Proyecto 2
MANUAL TECNICO

RUDY AARÓN GOPAL MARROQUÍN GARCIA

201903872

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Guatemala

2021

Requerimientos mínimos del sistema

Para el uso correcto de la aplicación es necesario los siguientes componentes y programas:

- Memoria ram de 2gb o superior
- Espacio disponible en disco de 500 mb
- Angular version 11.2.9
- Typescript 4.2.4 o superior
- Nodejs 6.14.7
- graphviz version 2.38.0 (20140413.2041)
- Sistema operative Windows 10 x64

Detalles del equipo utilizado.

Fabricante:	Acer
Modelo:	
Procesador:	Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz
Memoria instalada (RAM):	8.00 GB (7.85 GB utilizable)
Tipo de sistema:	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil:	La entrada táctil o manuscrita no está disponible para esta pantalla

Paradigma utilizado en el programa.

Para la creación de la aplicación se utilizó un paradigma orientado a objetos debido a que por el tipo de funcionamiento que se requería era el más óptimo para la realización de la aplicación, las clases utilizadas en la aplicación fueron necesarias para crear contenedores de datos específicos para su manejo de una manera más sencilla y optima, todos los árboles, expresiones regulares y conjuntos fueron registrados en nodos con sus respectivos datos y almacenados en listas de estos mismos tipos. Para la creación de los árboles se utilizó una metodología ascendente realizando todo el proceso inicialmente desde sus hojas hasta llegar a la raíz.

Explicación del lenguaje a utilizar.

La aplicación como es un interprete puede realizar distintas operaciones aritméticas, operaciones lógicas, ciclos, funciones, declaración y llamado de variables, métodos, todo programa para poder funcionar deberá de tener una función exec la cual es la función principal de la ejecución de la aplicación, se explicará la estructura de cada cosa.

1. **Operaciones aritméticas:** Las operaciones aritméticas que se pueden realizar son +, -, *, /, ^, % y todas poseen la misma estructura la cual es "EXP OPERACIÓN EXP" siendo exp cualquier valor u otra operación posible, también existe la operación de negación la cual si cambia su estructura por "MENOS EXP" es la única que utiliza esta estructura.
2. **Operaciones Lógicas:** Las operaciones lógicas que puede ser utilizadas son ==, ¡=, ¡ las cuales poseen la siguiente estructura "EXP LOGICA EXP" exceptuando el NOT la cual posee la estructura "NOT EXP".
3. **Ciclos**
    a. **FOR:** la estructura de un for es la siguiente "FOR (DECLARACION / ASIGNACION ; CONDICION ; ACTUALIZACION) { INSTRUCIONES }"
    b. **WHILE:** la estructura de un while es la siguiente "WHILE(CONDICION){INSTRUCCIONES}"
    c. **DO WHILE:** la estructura de un while es la siguiente "DO {INSTRUCIONES} WHILE(CONDICION)"
4. **Nativas:** existen diferentes funciones nativas con la estructura "FUNCION(EXP)" en las cuales podemos mencionar, toString, length, round, truncate, toUpper, toLower
5. **Funciones y métodos**
    a. **funciones:** los métodos tienen la siguiente estructura "TIPO ID (PARAMETROS){ INSTRUCIONES }"
    b. **métodos:** los métodos tienen la siguiente estructura "VOID ID (PARAMETROS) { INSTRUCCIONES } "
6. **Exec:** exec es una forma de llamar una función y a la vez indicara cual es nuestra función principal a ejecutar, solo puede existir uno por terminal "EXEC ID (PARAMETROS) "
7. **LLAMADAS:** para llamar una función o un método se utiliza la siguiente estructura " ID (PARAMETROS)"

Para explicar de mejor manera esta estructura vamos a ver la gramática, la cual fue generada con la aplicación JISON primero explicaremos las expresiones y símbolos que se podrán utilizar

- **CARACTERES QUE SE IGNORAN**

```
"//".*                  {}
[/][*][^*]*[*]+([^/*][^*]*[*]+)*[/]       {} //comentario multilinea
[ \r\t]+                {} //tabulador y retroceso
\n                      {} //enter
\s+                     {} //espacios
```

- **CARACTERES Y CADENAS**

Para estos dos casos fue utilizados los CASOS de Jison los cuales nos permiten cambiar de expresión regular a utilizar dependiendo de la entrada que tiene la expresión.

  - CARACTER

```
[']                 {Texto=""; this.begin("CARACTER");}
<CARACTER>[^'\\]"'"     {yytext = yytext.substr(0,yyleng-
1); this.popState(); return 'CARACTER';}
<CARACTER>"\\n'"        {yytext = '\n'; this.popState(); return 'CARACTER';}
<CARACTER>"\\t'"        {yytext = "\t"; this.popState(); return 'CARACTER';}
<CARACTER>"\\r'"        {yytext = "\r"; this.popState(); return 'CARACTER';}
<CARACTER>"\\\"'"       {yytext = "\""; this.popState(); return 'CARACTER';}
<CARACTER>"\\''"        {yytext = "'"; this.popState(); return 'CARACTER';}
<CARACTER>"\\\\'"       {yytext = "\\"; this.popState(); return 'CARACTER';}
<CARACTER><<EOF>>       return "EOF_IN_CARACTER";
<CARACTER>[^'\\]*"'"    {this.popState(); return 'CARACTER_ERROR';}
```

  - CADENA

```
["]                 {Texto=""; this.begin("Cadena");}
<Cadena>[^"\\]+     {Texto+=yytext;}
<Cadena>"\\n"       {Texto+='\n';}
<Cadena>"\\t"       {Texto+="\t";}
<Cadena>"\\r"       {Texto+="\r";}
<Cadena>"\\\""      {Texto+="\"";}
<Cadena>"\\'"       {Texto+="\'";}
<Cadena>"\\\\"      {Texto+="\\";}
<Cadena><<EOF>>     return "EOF_IN_STRING";
<Cadena>["]         {yytext = Texto; this.popState(); return 'Cadena';
}
```

- **SIMBOLOS**

Todos estos son los símbolos que podemos utilizar.

```
"PRINT"                 return "PRINT";
";"                     return "PTCOMA";
":"                     return "DOSPT"
"."                     return "PT";
","                     return "COMA";
"("                     return "PARIZ";
")"                     return "PARDER";
"["                     return "CORIZ";
"]"                     return "CORDER";
"{"                     return "LLAVEIZ";
"}"                     return "LLAVEDER";
"++"                    return "PLUS";
"--"                    return "MIN";

"+"                     return "MAS";
"-"                     return "MENOS";
"*"                     return "POR";
"/"                     return "DIV";
"%"                     return "MOD";
"^"                     return "ELEV";
"!="                    return "DIFERENTE";
"=="                    return "IIGUAL";
">="                    return "MAYORIGUAL";
"<="                    return "MENORIGUAL";

"<"                     return "MENOR";
">"                     return "MAYOR";
"="                     return "IGUAL";

"&&"                    return "AND";
"||"                    return "OR";
"!"                     return "NOT";

"INT"                   return "INT";
"DOUBLE"                return "DOUBLE";
"BOOLEAN"               return "BOOLEAN";
"CHAR"                  return "CHAR";
"STRING"                return "STRING";
"TRUE"                  return "TRUE";
"FALSE"                 return "FALSE";

```

```
"?"                     return "TERNARIO";
"FOR"                   return "FOR";
"WHILE"                 return "WHILE";
"DO"                    return "DO";
"SWITCH"                return "SWITCH";
"IF"                    return "IF";
"ELSE"                  return "ELSE";
"NEW"                   return "NEW";
"RETURN"                return "RETURN";
"CONTINUE"              return "CONTINUE";
"BREAK"                 return "BREAK";
"LIST"                  return "LIST";
"ADD"                   return "ADD";
"CASE"                  return "CASE";
"DEFAULT"               return "DEFAULT";
"VOID"                  return "VOID";
"toLOWER"               return "LOWER";
"toUPPER"               return "UPPER";
"Length"                return "LENGTH";
"Truncate"              return "TRUNCATE";
"Round"                 return "ROUND";
"Typeof"                return "TYPEOF";
"toSTRING"              return "TOSTRING";
"toCharArray"           return "CHARARRAY";
"Exec"                  return "EXEC";


[A-Za-z]([A-Za-z]|[0-9]|[_])*  return "ID";
[0-9]+"."[0-9]+\b         return "DOBLE";
[0-9]+\b                  return "ENTERO";

<<EOF>>                 return 'EOF';
```

- ## Analizador sintáctico

  Se muestra el código utilizado para el analizador sintáctico, para la explicación de cada cosa ver el documento de GRAMATICA.

  - ### PRECEDENCIA DE OPERADORES

    Para asegurar el buen funcionamiento de la aplicación se colocaron las precedencias de operadores necesarias, las cuales nos indican el orden en que se deben ejecutar las cosas.

```
%left 'TERNARIO'
%left 'OR'
%left 'AND'
%right 'NOT'
%left 'IIGUAL' 'DIFERENTE','MENOR','MENORIGUAL','MAYOR'.'MAYORIGUAL'
%left 'MAS' 'MENOS'
%left 'POR' 'DIV' 'MOD'
%left 'ELEV'
%right UMENOS
%right FCAST
%left 'PLUS','MIN'
```

  - ### INSTRUCIONES

```
    INS
        : PRINT PARIZ EXP PARDER PTCOMA                {$$ = new Imprimir.default(this._$.first_line, this._$.first_column, $3); }
        | DECLARACION PTCOMA                           {$$ = $1}
        | ASIGNACION PTCOMA                            {$$ = $1}
        | FIF                                          {$$ = $1}
        | FWHILE                                       {$$ = $1}
        | FFOR                                         {$$ = $1}
        | FSWITCH                                      {$$ = $1}
        | INCREMENTO PTCOMA                            {$$ = new INC.default(this._$.first_line, this._$.first_column, $1);}
        | DECREMENTO PTCOMA                            {$$ = new DEC.default(this._$.first_line, this._$.first_column, $1);}
        | DOWHILE                                      {$$ = $1}
        | FUNCION                                      {$$ = $1}
        | LLAMADA PTCOMA                               {if($1){$$ = new LLAMADA.default(this._$.first_line, this._$.first_column, $1);}else{$$="";}}
        | FRETURN                                      {$$ = $1}
        | BREAK PTCOMA                                 {$$ = new BREAK.default(this._$.first_line, this._$.first_column);}
        | CONTINUE PTCOMA                              {$$ = new CONTINUE.default(this._$.first_line, this._$.first_column);}
        | FTERNARIO PTCOMA                             {$$ = $1}
        | ID PT ADD PARIZ EXP PARDER PTCOMA            {$$ = new ADD.default(this._$.first_line, this._$.first_column, $1, $5);}
        | error PTCOMA                                 {ArbolAST.num_error++;ArbolAST.errores.push(new Excepcion.default(ArbolAST.num_error, "Sintactico
    ", "No se esperaba  "+yytext+".", this._$.first_line, this._$.first_column));}
        | error LLAVEDER {ArbolAST.num_error++;ArbolAST.errores.push(new Excepcion.default(ArbolAST.num_error, "Sintactico", "No se esperaba  "+yytext+".
    ", this._$.first_line, this._$.first_column));}
    ;
```

DECLARACION

```
DECLARACION
    :FTIPO ID                                                   {$$ = new DECLARAR.default(this._$.first_line, this._$.first_column,$2, $1)}
    |FTIPO ID IGUAL EXP                                         {$$ = new DECLARAR.default(this._$.first_line, this._$.first_column,$2, $1,-1,-1, $4)}
    |FTIPO CORIZ CORDER ID IGUAL NEW FTIPO CORIZ EXP CORDER      {$$ = new DECLARAR.default(this._$.first_line, this._$.first_column,$4, $1,$9,-1,undefined,$7)}
    |LIST MENOR FTIPO MAYOR ID                                  {$$ = new DECLARAR.default(this._$.first_line, this._$.first_column,$5, $2, $4,-1,0)}
    |FTIPO CORIZ CORDER ID IGUAL LLAVEIZ L_EXP LLAVEDER         {$$ = new DECLARAR.default(this._$.first_line, this._$.first_column,$4, $1,new Literal.default(this._$.first_line,
this._$.first_column,$7.length,Tipo.tipos.ENTERO),-1, $7)}
    |LIST MENOR FTIPO MAYOR ID IGUAL NEW LIST MENOR FTIPO MAYOR  {$$ = new DECLARAR.default(this._$.first_line, this._$.first_column,$5, $3, -
1,new Literal.default(this._$.first_line, this._$.first_column,0,Tipo.tipos.ENTERO),undefined,$10)}
    |LIST MENOR FTIPO MAYOR ID IGUAL EXP                        {$$ = new DECLARAR.default(this._$.first_line, this._$.first_column,$5, $3,-
1,new Literal.default(this._$.first_line, this._$.first_column,0,Tipo.tipos.ENTERO),$7)}
    ;
```

FOR

```
FFOR
    :FOR PARIZ DECLARACION PTCOMA EXP PTCOMA ACTUALIZACION  LLAVEIZ LINS LLAVEDER    {$$ = new FOR.default(this._$.first_line, this._$.first_column, $3, $5, $7, $9, "DEC");}
    |FOR PARIZ ASIGNACION PTCOMA EXP PTCOMA ACTUALIZACION  LLAVEIZ LINS LLAVEDER     {$$ = new FOR.default(this._$.first_line, this._$.first_column, $3, $5, $7, $9, "ASIG");}
    |FOR PARIZ ASIGNACION PTCOMA EXP PTCOMA ACTUALIZACION  LLAVEIZ LLAVEDER          {$$ = new FOR.default(this._$.first_line, this._$.first_column, $3, $5, $7, [], "ASIG");}
    |FOR PARIZ DECLARACION PTCOMA EXP PTCOMA ACTUALIZACION  LLAVEIZ LLAVEDER         {$$ = new FOR.default(this._$.first_line, this._$.first_column, $3, $5, $7, [], "DEC");}
    |FOR error LLAVEDER                                             {console.log("AQUI"); ArbolAST.num_error++; ArbolAST.errores.push(new Excepcion.default(ArbolAST.num_error, "Si
ntactico", "No se esperaba  "+yytext+".", this._$.first_line, this._$.first_column));}
    ;
```

WHILE

```
FWHILE
    :WHILE PARIZ EXP PARDER LLAVEIZ LINS LLAVEDER             {$$ = new WHILE.default(this._$.first_line, this._$.first_column, $3, $6);}
    |WHILE PARIZ EXP PARDER LLAVEIZ LLAVEDER                  {$$ = new WHILE.default(this._$.first_line, this._$.first_column, $3, []);}
    |WHILE error LLAVEDER                                     {ArbolAST.num_error++; ArbolAST.errores.push(new Excepcion.default(ArbolAST.num_error,
"Sintactico", "No se esperaba  "+yytext+".", this._$.first_line, this._$.first_column));}
    ;
```

DO WHILE

```
DOWHILE
    :DO LLAVEIZ LINS LLAVEDER WHILE PARIZ EXP PARDER PTCOMA             {$$ = new DOWHILE.default(this._$.first_line, this._$.first_column, $7, $3);}
    |DO LLAVEIZ LLAVEDER WHILE PARIZ EXP PARDER PTCOMA                  {$$ = new DOWHILE.default(this._$.first_line, this._$.first_column, $7, []);}
    |DO error PTCOMA                                                   {ArbolAST.num_error++; ArbolAST.errores.push(new Excepcion.default(ArbolAST.n
um_error, "Sintactico", "No se esperaba  "+yytext+".", this._$.first_line, this._$.first_column));}
    ;
```

IF, ELSE, ELSE IF

```
FIF

    :IF PARIZ EXP PARDER LLAVEIZ LINS LLAVEDER                              {$$ = new IF.default(this._$.first_line, this._$.first_column, $3, $6)}

    |IF PARIZ EXP PARDER LLAVEIZ LLAVEDER                                  {$$ = new IF.default(this._$.first_line, this._$.first_column, $3, [])}

    |IF PARIZ EXP PARDER LLAVEIZ LINS LLAVEDER ELSE FIF                    {$$ = new IF.default(this._$.first_line, this._$.first_column, $3, $6, undefined, $9)}

    |IF PARIZ EXP PARDER LLAVEIZ LINS LLAVEDER ELSE LLAVEIZ LINS LLAVEDER  {$$ = new IF.default(this._$.first_line, this._$.first_column, $3, $6, $10)}

    |IF PARIZ EXP PARDER LLAVEIZ LLAVEDER ELSE LLAVEIZ LLAVEDER            {$$ = new IF.default(this._$.first_line, this._$.first_column, $3, [], [])}

    |IF error LLAVEDER                                                     {ArbolAST.num_error++; ArbolAST.errores.push(new Excepcion.default(ArbolAST.num_error, "Sintactico", "
No se esperaba  "+yytext+".", this._$.first_line, this._$.first_column));}

;
```

ASIGNACION

```
ASIGNACION

    :ID IGUAL EXP                                {$$ = new ASIGNAR.default(this._$.first_line, this._$.first_column, $1,-1, $3);}

    |ID CORIZ CORIZ EXP CORDER CORDER IGUAL EXP  {$$ = new ASIGNAR.default(this._$.first_line, this._$.first_column, $1,$4, $8,"LIST");}

    |ID CORIZ EXP CORDER IGUAL EXP               {$$ = new ASIGNAR.default(this._$.first_line, this._$.first_column, $1,$3, $6,"VECTOR");}

;
```

FUNCION

```
FUNCION

    :FTIPO ID PARIZ PARDER LLAVEIZ LINS LLAVEDER              {$$ = ""; ArbolAST.FUNCIONES.push(new FUNC.default(this._$.first_line, this._$.first_column,$1, $2, $6));}

    |FTIPO ID PARIZ PARDER LLAVEIZ LLAVEDER                   {$$ = ""; ArbolAST.FUNCIONES.push(new FUNC.default(this._$.first_line, this._$.first_column,$1, $2, []));}

    |FTIPO ID PARIZ PARAMETROS PARDER LLAVEIZ LINS LLAVEDER   {$$ = ""; ArbolAST.FUNCIONES.push(new FUNC.default(this._$.first_line, this._$.first_column,$1, $2, $7, $4));}

    |FTIPO ID PARIZ PARAMETROS PARDER LLAVEIZ LLAVEDER        {$$ = ""; ArbolAST.FUNCIONES.push(new FUNC.default(this._$.first_line, this._$.first_column,$1, $2, [], $4));}

    |VOID ID PARIZ PARAMETROS PARDER LLAVEIZ LINS LLAVEDER    {$$ = ""; ArbolAST.FUNCIONES.push(new FUNC.default(this._$.first_line, this._$.first_column,new Tipo.default(Tipo.tipos.ENTERO), $2, $7, $4,true));}

    |VOID ID PARIZ PARDER LLAVEIZ LINS LLAVEDER               {$$ = ""; ArbolAST.FUNCIONES.push(new FUNC.default(this._$.first_line, this._$.first_column,new Tipo.default(Tipo.tipos.ENTERO), $2, $6, undefined,true));}

    |VOID ID PARIZ PARAMETROS PARDER LLAVEIZ LLAVEDER         {$$ = ""; ArbolAST.FUNCIONES.push(new FUNC.default(this._$.first_line, this._$.first_column,new Tipo.default(Tipo.tipos.ENTERO), $2, [], $4,true));}

    |VOID ID PARIZ PARDER LLAVEIZ LLAVEDER                   {$$ = ""; ArbolAST.FUNCIONES.push(new FUNC.default(this._$.first_line, this._$.first_column,new Tipo.default(Tipo.tipos.ENTERO), $2, [], undefined,true));}

    |VOID error LLAVEDER                                     {ArbolAST.num_error++; ArbolAST.errores.push(new Excepcion.default(ArbolAST.num_error, "Sintactico", "No se esperaba  "+yytext+".", this._$.first_line, this._$.first_column));}

;
```

PARAMETROS

```
PARAMETROS

    :PARAMETROS COMA FTIPO ID        {$$ = []; $$ = $1; $$.push(new DECLARAR.default(this._$.first_line, this._$.first_column,$4, $3));}

    |FTIPO ID                        {$$ = []; $$.push(new DECLARAR.default(this._$.first_line, this._$.first_column,$2, $1));}

;
```

- ## SWITCH

```
FSWITCH
    :SWITCH PARIZ EXP PARDER LLAVEIZ LCASOS DEFAULT DOSPT LINS LLAVEDER            {$$ = new SWITCH.default(this._$.first_line, this._$.first_column,$3,$6, $9)}
    |SWITCH PARIZ EXP PARDER LLAVEIZ LCASOS LLAVEDER                              {$$ = new SWITCH.default(this._$.first_line, this._$.first_column,$3,$6, undefined)}
    |SWITCH PARIZ EXP PARDER LLAVEIZ DEFAULT DOSPT LINS LLAVEDER                  {$$ = new SWITCH.default(this._$.first_line, this._$.first_column,$3,undefined, $8)}
    |SWITCH error PARDER                                                         {ArbolAST.num_error++; ArbolAST.errores.push(new Excepcion.default(ArbolAST.num_error, "Sintactico", "
No se esperaba  "+yytext+".", this._$.first_line, this._$.first_column));}
;
```

- ## LLAMADA

```
LLAMADA
    :ID PARIZ L_EXP PARDER              {$$ = new FUNCION.default(this._$.first_line, this._$.first_column, $1, $3);}
    |ID PARIZ PARDER                    {$$ = new FUNCION.default(this._$.first_line, this._$.first_column, $1, undefined);}
    |EXEC ID PARIZ L_EXP PARDER         {$$ = undefined; ArbolAST.exec.push(new FUNCION.default(this._$.first_line, this._$.first_column, $2, $4));}
    |EXEC ID PARIZ PARDER               {$$ = undefined; ArbolAST.exec.push(new FUNCION.default(this._$.first_line, this._$.first_column, $2, undefin
ed));}
;
```

- ## NATIVAS

```
NATIVAS
    :LENGTH PARIZ EXP PARDER           {$$ = new NATIVAS.default(this._$.first_line, this._$.first_column, $1, $3);}
    |TRUNCATE PARIZ EXP PARDER         {$$ = new NATIVAS.default(this._$.first_line, this._$.first_column, $1, $3);}
    |ROUND PARIZ EXP PARDER            {$$ = new NATIVAS.default(this._$.first_line, this._$.first_column, $1, $3);}
    |TYPEOF PARIZ EXP PARDER           {$$ = new NATIVAS.default(this._$.first_line, this._$.first_column, $1, $3);}
    |TOSTRING PARIZ EXP PARDER         {$$ = new NATIVAS.default(this._$.first_line, this._$.first_column, $1, $3);}
    |CHARARRAY PARIZ EXP PARDER      {$$ = new NATIVAS.default(this._$.first_line, this._$.first_column, $1, $3);}
;
```

- ## VALORES

```
LISTAVALORES
    :ENTERO             {$$ = new Literal.default(this._$.first_line, this._$.first_column, $1, Tipo.tipos.ENTERO)}
    |DOBLE              {$$ = new Literal.default(this._$.first_line, this._$.first_column, $1, Tipo.tipos.DOBLE)}
    |CARACTER           {$$ = new Literal.default(this._$.first_line, this._$.first_column, $1, Tipo.tipos.CARACTER)}
    |Cadena             {$$ = new Literal.default(this._$.first_line, this._$.first_column, $1, Tipo.tipos.CADENA)}
    |TRUE               {$$ = new Literal.default(this._$.first_line, this._$.first_column, $1, Tipo.tipos.BOOLEANO)}
    |FALSE              {$$ = new Literal.default(this._$.first_line, this._$.first_column, $1, Tipo.tipos.BOOLEANO)}
;
```

- ## EXEC

```
    |EXEC ID PARIZ L_EXP PARDER         {$$ = undefined; ArbolAST.exec.push(new FUNCION.default(this._$.first_line, this._$.first_column, $2, $4));}
    |EXEC ID PARIZ PARDER               {$$ = undefined; ArbolAST.exec.push(new FUNCION.default(this._$.first_line, this._$.first_column, $2, undefin
ed));}
```

EXPRESIONES

```
EXP

    :EXP MAS EXP                        {$$ = new Aritmetica.default(Aritmetica.OperadorAritmetico.SUMA,this._$.first_line, this._$.first_column, 0
                                                , Tipo.tipos.ENTERO, $1, $3)}
    |EXP MENOS EXP                      {$$ = new Aritmetica.default(Aritmetica.OperadorAritmetico.RESTA,this._$.first_line, this._$.first_column, 0
                                                , Tipo.tipos.ENTERO, $1, $3)}
    |EXP POR EXP                        {$$ = new Aritmetica.default(Aritmetica.OperadorAritmetico.MULTIPLICACION,this._$.first_line, this._$.first_column, 0
                                                , Tipo.tipos.ENTERO, $1, $3)}
    |EXP DIV EXP                        {$$ = new Aritmetica.default(Aritmetica.OperadorAritmetico.DIVISION,this._$.first_line, this._$.first_column, 0
                                                , Tipo.tipos.ENTERO, $1, $3)}
    |EXP MOD EXP                        {$$ = new Aritmetica.default(Aritmetica.OperadorAritmetico.MODULO,this._$.first_line, this._$.first_column, 0
                                                , Tipo.tipos.ENTERO, $1, $3)}
    |EXP ELEV EXP                       {$$ = new Aritmetica.default(Aritmetica.OperadorAritmetico.POTENCIA,this._$.first_line, this._$.first_column, 0
                                                , Tipo.tipos.ENTERO, $1, $3)}
    |MENOS EXP %prec UMENOS             {$$ = new Aritmetica.default(Aritmetica.OperadorAritmetico.RESTA,this._$.first_line, this._$.first_column, 0
                                                , Tipo.tipos.ENTERO, $2)}
    |PARIZ EXP PARDER                   {$$ = $2}
    |LISTAVALORES                       {$$ = $1}
    |EXP MENOR EXP                      {$$ = new Condicion.default(this._$.first_line, this._$.first_column, 0, "<", $1, $3);}
    |EXP MAYOR EXP                      {$$ = new Condicion.default(this._$.first_line, this._$.first_column, 0, ">", $1, $3);}
    |EXP DIFERENTE EXP                  {$$ = new Condicion.default(this._$.first_line, this._$.first_column, 0, "!=", $1, $3);}
    |EXP IIGUAL EXP                     {$$ = new Condicion.default(this._$.first_line, this._$.first_column, 0, "==", $1, $3);}
    |EXP MAYORIGUAL EXP                 {$$ = new Condicion.default(this._$.first_line, this._$.first_column, 0, ">=", $1, $3);}
    |EXP MENORIGUAL EXP                 {$$ = new Condicion.default(this._$.first_line, this._$.first_column, 0, "<=", $1, $3);}
    |EXP AND EXP                        {$$ = new Condicion.default(this._$.first_line, this._$.first_column, 0, "&&", $1, $3);}
    |EXP OR EXP                         {$$ = new Condicion.default(this._$.first_line, this._$.first_column, 0, "||", $1, $3);}
    |NOT EXP                            {$$ = new Condicion.default(this._$.first_line, this._$.first_column, 0, "!", $2);}
    |CAST                               {$$ = $1}
    |FTERNARIO                          {$$ = $1}
    |INCREMENTO                         {$$ = $1}
    |DECREMENTO                         {$$ = $1}
    |NATIVAS                            {$$ = $1}
    |FTOLOWER                           {$$ = $1}
    |FTOUPPER                           {$$ = $1}
    |ID                                 {$$ = new Variable.default(this._$.first_line, this._$.first_column, $1);}
    |LLAMADA                            {$$ = $1}
    |ID CORIZ  EXP CORDER               {$$ = new Vector.default(this._$.first_line, this._$.first_column, $1, $3, "VECTOR");}
    |ID CORIZ CORIZ EXP CORDER CORDER   {$$ = new Vector.default(this._$.first_line, this._$.first_column, $1, $4, "LIST");}
;
```

- **Explicación de clases utilizadas**

  - **EXPRESIONES**

Aritmética	Esta clase devuelve una litera, es utilizada para realizar todas las operaciones aritméticas
Casteo	Esta clase devuelve una literal, es utilizada para realizar los distintos casteos
Condición	Esta clase devuelve una litera, es utilizada para realizar diferentes tipos de operaciones lógicas y comparaciones entre literales
Decremento	Esta clase devuelve una literal, disminuye en 1 una variable
Expresión	Clase abstracta que utilizan todas las expresiones
Función	Esta clase es utilizada para la ejecución de una función devolviendo una literal con el tipo indicado para la función
Incremento	Esta clase devuelve una literal, aumenta en 1 una variable
Literal	Clase que almacenara un valor
Nativas	Esta función se encarga de ejecutar todas las funciones nativas de la aplicación
Ternario	Esta clase se encarga de ejecutar una operación ternaria
Tolower	Esta Clase devuelve un string con todos sus caracteres en minúsculas
Toupper	Esta clase devuelve un string con todas sus clases en mayúsculas

Variable	Esta clase almacena una literal
vector	Esta clase almacena una literal con un valor de lista

- **INSTRUCIONES**

Add	Esta clase añade un nuevo valor a una variable de tipo lista
Asignar	Esta clase asigna un nuevo valor a las variables
Break	Esta clase rompe el funcionamiento de un ciclo
Continue	Esta clase se salta un paso de un ciclo sin ejecutar las instrucciones que le sigan
Declarar	Esta clase se utiliza para crear nuevas variables, listas y vectores
Decremento	Esta clase ejecuta la acción decremento
Dowhile	Esta clase realiza la instrucción do while recibe una condición e instrucciones
For	Esta clase realiza la instrucción for, recibe una condición una declaración y una lista de instrucciones
Función	Esta clase registra una nueva función
If	Esta clase es utilizada para condiciones if, else y else if
Imprimir	Esta clase imprime una variable
Incrementar	Esta clase ejecuta la acción incrementar
Llamada	Esta clase es utilizada para ejecutar una función

Return	Esta clase devuelve un valor se puede utilizar únicamente dentro de una función
Switch	Esta clase ejecuta un switch, el cual contiene múltiples casos e instrucciones
while	Esta clase ejecuta el ciclo while, recibe una condición e instrucciones a realizar.