Gramática

CompScript

Diego Abraham Robles Meza

Universidad de san carlos de guatemala

Organización de lenguajes y compiladores 1

Sección “B”

**Terminales:**

“true” = {TRUE}

“false” = {FALSE}

“print” = {PRINT}

“println” = {PRINTLN}

“typeof” = {TYPE\_OF}

“toString” = {TO\_STRING}

“toCharArray” = {TO\_CHAR\_ARRAY}

“length” = {LENGTH}

“toLower” = {TO\_LOWER}

“toUpper” = {TO\_UPPER}

“round” = {ROUND}

“if” = {IF}

“else” = {ELSE}

“do” = {DO}

“while” = {WHILE}

“switch” = {SWITCH}

“case” = {CASE}

“default” = {DEFAULT}

“for” = {FOR}

“break” = {BREAK}

“continue” = {CONTINUE}

“return” = {RETURN}

“new” = {NEW}

“void” = {VOID}

“run” = {RUN}

“int” = {INT]

“double” = {DOUBLE}

“char” = {CHAR}

“boolean” = {BOOLEAN}

“string” = {STRING}

“vector” = {VECTOR}

\d+\.\d+ = {DECIMAL}

\d+ = {ENTERO}

\’.\’ = {CARACTER}

(\"[^\"]\*\") = {CADENA}

([a-zA-Z])[a-zA-Z0-9\_]\* = {IDENTIFICADOR}

“(“ = {PAR\_ABRE}

“)” = {PAR\_CIERRA}

“{“ = {LL\_ABRE}

“}” = {LL\_CIERRA}

“[“ = {COR\_ABRE}

“]” = {COR\_CIERRA}

“==” = {IGUAL\_IGUAL}

“<=” = {MENOR\_IGUAL}

“<” = {MENOR}

“>=” = {MAYOR\_IGUAL}

“>” = {MAYOR}

“!=” = {DIFERENTE}

“=” = {IGUAL}

“?” = {QUESTION}

“||” = {OR}

“&&” = {AND}

“!” = {NOT}

“,” = {COMA}

“:” = {DOS\_PUNTOS}

“;” = {PUNTO\_Y\_COMA}

“++” = {MAS\_MAS}

“+” = {MAS}

“- -“ = {MENOS\_MENOS}

“-“ = {MENOS}

“\*” = {POR}

“/” = {DIVIDIR}

“^” = {POTENCIA}

“%” = {MODULO}

**No terminales:**

ini

instrucciones

inicio

declaración

declaracion\_multiple

asignacion

print

listaExpr

if

else

switch

lista\_case

case

statement\_case

default

while

for

do\_while

statement

function

tipo\_dato

parametros

tipos

llamada\_funcion

return

incremento\_decremento

ternario

casteos

declaracion\_vectores

vector\_una\_dimension

vector\_dos\_dimensiones

acceso\_vectores

modificacion\_vectores

type\_of

to\_string

to\_char\_array

length

to\_lower

to\_upper

round

expresion

**Producciones:**

**Producción inicial:**

**ini** ::= **instrucciones** EOF

**Producción de instrucciones:**

Esta producción es una de las principales, ya que puede contener una o muchas instrucciones de otros tipos, sirve para encapsular varias sentencias dentro de algún bloque como un “if”, “else” o el cuerpo de alguna función.

**instrucciones** ::= **instrucciones** **inicio**

| **inicio**

**Producción de inicio:**

Esta producción es la que se encarga de recibir una instrucción, la producción de instrucciones depende de esta y de esta derivan todas las demás producciones.

**inicio** ::= **asignacion** PUNTO\_Y\_COMA

| **declaracion** PUNTO\_Y\_COMA

| **print** PUNTO\_Y\_COMA

| **if**

| **while**

| **do\_while**

| **for**

| **switch**

| **function**

| **llamada\_funcion** PUNTO\_Y\_COMA

| **incremento\_decremento** PUNTO\_Y\_COMA

| BREAK PUNTO\_Y\_COMA

| CONTINUE PUNTO\_Y\_COMA

| **return** PUNTO\_Y\_COMA

| **declaracion\_vectores** PUNTO\_Y\_COMA

| **modificacion\_vectores** PUNTO\_Y\_COMA

**Producción de declaración:**

Esta producción es la que se encarga de recibir una entrada como las siguientes:

tipo\_dato id

tipo\_dato lista\_ids

tipo\_dato id = valor

tipo\_dato lista\_ids = valor

tipo\_dato id = casteo

tipo\_dato lista\_ids = casteo

en donde: tipo\_dato puede ser int, double, char, boolean, string.

**declaracion** ::= INT **declaracion\_multiple**

| DOUBLE **declaracion\_multiple**

| CHAR **declaracion\_multiple**

| BOOLEAN **declaracion\_multiple**

| STRING **declaracion\_multiple**

| INT **declaracion\_multiple** IGUAL **expresion**

| DOUBLE **declaracion\_multiple** IGUAL **expresion**

| CHAR **declaracion\_multiple** IGUAL **expresion**

| BOOLEAN **declaracion\_multiple** IGUAL **expresion**

| STRING **declaracion\_multiple** IGUAL **expresion**

| INT **declaracion\_multiple** IGUAL **casteos**

| DOUBLE **declaracion\_multiple** IGUAL **casteos**

| CHAR **declaracion\_multiple** IGUAL **casteos**

| BOOLEAN **declaracion\_multiple** IGUAL **casteos**

| STRING **declaracion\_multiple** IGUAL **casteos**

**declaracion\_multiple** ::= **declaracion\_multiple** COMA IDENTIFICADOR

| IDENTIFICADOR

**Producción de asignación:**

Esta producción se encarga de reconocer una entrada como las siguientes:

Id = valor

**asignacion** ::= IDENTIFICADOR IGUAL **expresion**

| IDENTIFICADOR

**Producción de Print y Println:**

Esta producción reconoce una entrada como la siguiente:

Print (valor)

Print(lista\_valores)

Println (valor)

Println(lista\_valores)

**print** ::= PRINT PAR\_ABRE **listaExpr** PAR\_CIERRA

| PRINTLN PAR\_ABRE **listaExpr** PAR\_CIERRA

**listaExpr** ::= **listaExpr** COMA **expresion**

| **expresion**

**Producción de IF y Else:**

Esta producción recibe una entrada como convencionalmente viene un if, teniendo las siguientes posibilidades de ocurrencia:

If (condición) { statement } else { statement }

If (condición) { statement }

If (condición) { statement } else if (condicion) { statement }

Tomar en cuenta que un if y else pueden tener anidados muchos ifs en conjunto con else y que la condición a evaluar debe ser de tipo Booleana.

**if** ::= IF PAR\_ABRE **expresion** PAR\_CIERRA **statement** **elsE**

**elsE** ::= ELSE **statement**

| ELSE **if**

|

**Producción de Switch:**

Esta producción acepta una entrada como la del switch convencionalmente vista en el lenguaje de programación JAVA, en donde a partir de una condición a evaluar se compara con los distintos casos que existen y en caso de que ninguna opción cumpla se ejecutará la opción por defecto si es que existe:

Switch (opción) { lista\_casos default }

**switch** ::= SWITCH PAR\_ABRE **expresion** PAR\_CIERRA LL\_ABRE **lista\_case** **default** LL\_CIERRA

| SWITCH PAR\_ABRE **expresion** PAR\_CIERRA LL\_ABRE **default** LL\_CIERRA

**lista\_case** ::= **lista\_case** **case**

| **case**

**case** ::= CASE **expresion** DOSPUNTOS **statement\_case**

**statement\_case** ::= **instrucciones**

|

**default** ::= DEFAULT DOS\_PUNTOS **statement\_case**

|

**Producción de While y Do While:**

Esta producción al igual que la del switch sigue la sintaxis de un bucle while, similar a la de los lenguajes más comunes:

While (condición) { statement }

Do { statement } while (condición)

Tomar en cuenta que la condición a evaluar debe ser de tipo Booleana.

**while** ::= WHILE PAR\_ABRE **expresion** PAR\_CIERRA **statement**

**do\_while** ::= DO **statement** WHILE PAR\_ABRE **expresion** PAR\_CIERRA

**Producción de For:**

Esta producción se encarga de recibir una entrada como las del bucle for convencional, en donde se tienen 3 parámetros:

For (declaracion/asignacion ; condición ; actualización ) { statement }

**for** ::= FOR PAR\_ABRE **declaracion** PUNTO\_Y\_COMA **expresion** PUNTO\_Y\_COMA **incremento\_decremento** PAR\_CIERRA **statement**

| FOR PAR\_ABRE **declaracion** PUNTO\_Y\_COMA **expresion** PUNTO\_Y\_COMA **asignacion** PAR\_CIERRA **statement**

| FOR PAR\_ABRE **asignacion** PUNTO\_Y\_COMA **expresion** PUNTO\_Y\_COMA **incremento\_decremento** PAR\_CIERRA **statement**

| FOR PAR\_ABRE **asignacion** PUNTO\_Y\_COMA **expresion** PUNTO\_Y\_COMA **asignacion** PAR\_CIERRA **Statement**

**Producción Statement (Cuerpo):**

Esta producción es una de las más importantes, ya que es la encargada de estableces todo el cuerpo de las instrucciones evaluadas dentro de las estructuras más conocidas (if, else, while, switch, for, etc.). Se establece de la siguiente forma:

{ instrucciones }

{ }

**statement** ::= LL\_ABRE **instrucciones** LL\_CIERRA

| LL\_ABRE LL\_CIERRA

**Producción de Funciones:**

Esta producción es la encargada del reconocimiento de funciones, ya sea que tengan un tipo concreto(void, int, double, char, boolean, string) o que no tengan uno establecido(en este caso se le asigna void por defecto).

Id () : tipo { statement }

Id () { statement }

Id ( parametros ) : tipo { statement }

Id ( parametros ) { statement }

Tomar en cuenta que los parámetros al crear la función tienen que ser del mismo tipo al ser llamada esta función y se ingresan en el orden que los pide la función.

**function** ::= IDENTIFICADOR PAR\_ABRE PAR\_CIERRA DOS\_PUNTOS **tipo\_dato** **statement**

| IDENTIFICADOR PAR\_ABRE **parametros** PAR\_CIERRA DOS\_PUNTOS **tipo\_dato** **statement**

| IDENTIFICADOR PAR\_ABRE PAR\_CIERRA **statement**

| IDENTIFICADOR PAR\_ABRE **parametros** PAR\_CIERRA **statement**

**tipo\_dato** ::= INT

| DOUBLE

| CHAR

| BOOLEAN

| STRING

| VOID

**Producción de Parámetros:**

En la creación de las funciones puede que estas tengan parámetros dentro de los paréntesis. Esta producción es la que se encarga de reconocer todos los parámetros que reciben las funciones y pueden ser de la siguiente forma:

Tipo id

Tipo [ ] id

Tipo id [ ]

Tipo [ ] [ ] id

Tipo id [ ] [ ]

**parametros** ::= **parametros** COMA **tipos** IDENTIFICADOR

| **tipos** IDENTIFICADOR

| **parametros** COMA **tipos** COR\_ABRE COR\_CIERRA IDENTIFICADOR

| **tipos** COR\_ABRE COR\_CIERRA IDENTIFICADOR

| **parametros** COMA **tipos** IDENTIFICADOR COR\_ABRE COR\_CIERRA

| **tipos** IDENTIFICADOR COR\_ABRE COR\_CIERRA

| **parametros** COMA **tipos** COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR

| **tipos** COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR

| **parametros** COMA **tipos** IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA

| **tipos** IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA

**tipos** ::= INT

| DOUBLE

| CHAR

| BOOLEAN

| STRING

| VECTOR

**Producción de Llamada Función:**

Esta producción se encarga de reconocer las llamadas que se realicen a funciones dentro del programa, se hacen de la siguiente manera:

Id ();

Id( parámetros ) ;

Tener en cuenta que si una función es llamada y la cantidad de parámetros es incorrecta o si el tipo no es del mismo tipo que recibe la función al ser declarada esto ocasionará un error.

**llamada\_funcion** ::= IDENTIFICADOR PAR\_ABRE PAR\_CIERRA

| IDENTIFICADOR PAR\_ABRE **ListaExpr** PAR\_CIERRA

| RUN IDENTIFICADOR PAR\_ABRE PAR\_CIERRA

| RUN IDENTIFICADOR PAR\_ABRE **ListaExpr** PAR\_CIERRA

**Producción de Return:**

Esta producción es la que se encarga de reconocer la instrucción return dentro del lenguaje.

Return ;

Return valor;

**return** ::= RETURN

| RETURN **expresion**

**Producción de Incremento y Decremento:**

Esta producción se encarga de reconocer una entrada como la siguiente;

Id ++;

Id --;

**incremento\_decremento** ::= IDENTIFICADOR MAS\_MAS

| IDENTIFICADOR MENOS\_MENOS

**Producción de Operador Ternario:**

Esta producción se encarga de reconocer una entrada como la siguiente:

( condición ) ? valor : valor ;

**ternario** ::= **expresion** QUESTION **expresion** DOS\_PUNTOS **expresion**

**Producción de Casteos:**

Esta producción se encarga de reconocer una entrada como la siguiente:

( tipo ) valor;

**casteos** ::= PAR\_ABRE DOUBLE PAR\_CIERRA **expresion**

| PAR\_ABRE CHAR PAR\_CIERRA **expresion**

| PAR\_ABRE INT PAR\_CIERRA **expresion**

| PAR\_ABRE BOOLEAN PAR\_CIERRA **expresion**

| PAR\_ABRE STRING PAR\_CIERRA **expresion**

**Producción de Vectores(1 y 2 dimensiones):**

Esta producción genera otras dos producciones, la cuales sirven para reconocer entradas como las siguientes:

Tipo id [ ] = new tipo [ # ];

Tipo [ ] id = new tipo[ # ];

Tipo id [ ] = [n1, n2, n3, …];

Tipo [ ] id = [n1, n2, n3, …];

Tipo id [ ] [ ] = new tipo [ # ] [ # ];

Tipo [ ] [ ] id = new tipo [ # ] [ # ];

Tipo id [ ] [ ] id = [[n1, n2, n3, …], [n1, n2, n3, …], …];

Tipo [ ] [ ] id = [[n1, n2, n3, …], [n1, n2, n3, …], …];

Donde # es el tamaño de las filas o columnas y este debe de ser de tipo entero(int).

**declaracion\_vectores** ::= **vector\_una\_dimension**

| **vector\_dos\_dimensiones**

**vector\_una\_dimension** ::= INT IDENTIFICADOR COR\_ABRE COR\_CIERRA IGUAL NEW INT COR\_ABRE **expresion** COR\_CIERRA

| INT IDENTIFICADOR COR\_ABRE COR\_CIERRA IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| DOUBLE IDENTIFICADOR COR\_ABRE COR\_CIERRA IGUAL NEW DOUBLE COR\_ABRE **expresion** COR\_CIERRA

| DOUBLE IDENTIFICADOR COR\_ABRE COR\_CIERRA IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| CHAR IDENTIFICADOR COR\_ABRE COR\_CIERRA IGUAL NEW CHAR COR\_ABRE **expresion** COR\_CIERRA

| CHAR IDENTIFICADOR COR\_ABRE COR\_CIERRA IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| CHAR IDENTIFICADOR COR\_ABRE COR\_CIERRA IGUAL **to\_char\_array**

| BOOLEAN IDENTIFICADOR COR\_ABRE COR\_CIERRA IGUAL NEW BOOLEAN COR\_ABRE **expresion** COR\_CIERRA

| BOOLEAN IDENTIFICADOR COR\_ABRE COR\_CIERRA IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| STRING IDENTIFICADOR COR\_ABRE COR\_CIERRA IGUAL NEW STRING COR\_ABRE **expresion** COR\_CIERRA

| STRING IDENTIFICADOR COR\_ABRE COR\_CIERRA IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| INT COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL NEW INT COR\_ABRE **expresion** COR\_CIERRA

| INT COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| DOUBLE COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL NEW DOUBLE COR\_ABRE **expresion** COR\_CIERRA

| DOUBLE COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| CHAR COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL NEW CHAR COR\_ABRE **expresion** COR\_CIERRA

| CHAR COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| CHAR COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL **to\_char\_array**

| BOOLEAN COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL NEW BOOLEAN COR\_ABRE **expresion** COR\_CIERRA

| BOOLEAN COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| STRING COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL NEW STRING COR\_ABRE **expresion** COR\_CIERRA

| STRING COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

**vector\_dos\_dimensiones** ::= INT IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IGUAL NEW INT COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA

| INT IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| DOUBLE IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IGUAL NEW DOUBLE COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA

| DOUBLE IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| CHAR IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IGUAL NEW CHAR COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA

| CHAR IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| BOOLEAN IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IGUAL NEW BOOLEAN COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA

| BOOLEAN IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| STRING IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IGUAL NEW STRING COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA

| STRING IDENTIFICADOR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| INT COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL NEW INT COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA

| INT COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| DOUBLE COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL NEW DOUBLE COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA

| DOUBLE COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| CHAR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL NEW CHAR COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA

| CHAR COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| BOOLEAN COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL NEW BOOLEAN COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA

| BOOLEAN COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

| STRING COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL NEW STRING COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA

| STRING COR\_ABRE COR\_CIERRA COR\_ABRE COR\_CIERRA IDENTIFICADOR IGUAL COR\_ABRE **ListaExpr** COR\_CIERRA

**Producción de Acceso a vectores(1 y 2 dimensiones):**

En esta producción se reconocen las entradas que hacen referencia a la posición de un arreglo en cierta posición específica:

Id [ # ];

Id [ # ] [ # ];

Tener en cuenta que # debe de ser de tipo entero(int) de lo contrario se considera un error.

**acceso\_vectores** ::= IDENTIFICADOR COR\_ABRE **expresion** COR\_CIERRA

| IDENTIFICADOR COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA

**Producción de Modificación de vectores(1 y 2 dimensiones):**

**modificacion\_vectores** ::= IDENTIFICADOR COR\_ABRE **expresion** COR\_CIERRA IGUAL **expresion**

| IDENTIFICADOR COR\_ABRE **expresion** COR\_CIERRA COR\_ABRE **expresion** COR\_CIERRA IGUAL **expresion**

**Producciones de funciones nativas:**

Estas producciones siguen una sintaxis bastante parecida, la cual consta de los siguientes tokens:

Tipo\_funcion\_nativa ( expresion );

En donde Tipo\_funcion\_nativa puede ser:

* typeOf
* toString
* toCharArray
* length
* toLower
* toUpper
* Round

**type\_of** ::= TYPEOF PAR\_ABRE **expresion** PAR\_CIERRA

**to\_string** ::= TOSTRING PAR\_ABRE **expresion** PAR\_CIERRA

**to\_char\_array** ::= TOCHARARRAY PAR\_ABRE **expresion** PAR\_CIERRA

**length\_** ::= LENGTH PAR\_ABRE **expresion** PAR\_CIERRA

**to\_lower** ::= TOLOWER PAR\_ABRE **expresion** PAR\_CIERRA

**to\_upper** ::= TOUPPER PAR\_ABRE **expresion** PAR\_CIERRA

**round** ::= ROUND PAR\_ABRE **expresion** PAR\_CIERRA

**Producción de Expresión:**

En esta producción se reconocen todas las entradas que van a ser asignadas a diferentes funciones o variables:

**expresion** ::= MENOS **expresion**

| **expresion** MAS **expresion**

| **expresion** MENOS **expresion**

| **expresion** POR **expresion**

| **expresion** DIVIDIR **expresion**

| **expresion** POTENCIA **expresion**

| **expresion** MODULO **expresion**

| **expresion** IGUAL\_IGUAL **expresion**

| **expresion** DIFERENTE **expresion**

| **expresion** MAYOR\_IGUAL **expresion**

| **expresion** MENOR\_IGUAL **expresion**

| **expresion** MAYOR **expresion**

| **expresion** MENOR **expresion**

| PAR\_ABRE **expresion** PAR\_CIERRA

| **expresion** AND **expresion**

| **expresion** OR **expresion**

| NOT **expresion**

| CADENA

| ENTERO

| DECIMAL

| TRUE

| FALSE

| CARACTER

| IDENTIFICADOR

| **ternario**

| **incremento\_decremento**

| COR\_ABRE **ListaExpr** COR\_CIERRA

| **acceso\_vectores**

| **type\_of**

| **to\_string**

| **to\_char\_array**

| **length\_**

| **to\_lower**

| **to\_upper**

| **round**

| **llamada\_funcion**