GRAMATICA

Para comenzar se realizó el análisis léxico donde se definieron diferentes tokens para poder utilizarlos en el análisis léxico a continuación se mostrara un listado de todos los tokens definidos.

TOKEN	PATRON
Comentario Una Linea	[/][/].*
Comentario varias líneas	[/][*][^*]*[*]+([^/*][^*]*[*]+)*[/]
MAIN	“main”
VOID	“void”
PRINT	“print”
TOLOWER	“toLower”
TOUPPER	"toUpper"
LENGHT	"Lenght"
TRUNCATE	"Truncate"
ROUND	"Round"
TYPEOF	"Typeof"
TOSTRING	"toString"
TOCHARARRAY	"toCharArray"
IF	"if"
ELSE	"else"
SWITCH	"switch"
CASE	"case"
DEFAULT	"default"
WHILE	"while"
FOR	"for"
DO	"do"
BREAK	"break"
CONTINUE	"continue"
RETURN	"return"
NEW	"new"
ADD	'add'
KLENNE	?
DOSPUNTOS	:
PUNTOCOMA	;
PARIZQ	(
PARDER	)
LLAVEIZQ	{
LLAVERDER	}
CORIZQ	[
CORDER	]
COMA	,
PUNTO	.
RES_INT	Int

RES_DOUBLE	Double
RES_BOOLEAN	Boolean
RES_CHAR	Char
RES_STRING	String
INCREMENTO	++
DECREMENTO	--
MAS	+
MENOS	-
POR	*
DIVISION	/
POTENCIA	^
MODULO	%
IGUALAR	==
IGUAL	=
NOIGUAL	!=
MENORIGUAL	<=
MENOR	<
MAYORIGUAL	>=
MAYOR	>
OR	||
AND	&&
NOT	!
DOUBLE	[0-9]+("."[0-9]+)\b
INT	[0-9]+
BOOL	"true"|"false"
CHAR	(\'[^']\')
CADENA	(\"([^\"]|"\\\"")*\")
IDENTIFICADOR	[a-zA-Z_][a-zA-Z0-9_]*

Luego de definir todos los tokens que vendrían a ser los **terminales** de nuestra gramática

Luego tenemos los **no terminales**:

Init

Instrucciones

Instrucción

Main

Print

Expression

Primitivo

Declaration

Tipo

Accvar

Accvector

For

While

Dowhile

Cif

Celse

Ternario

Aritmética

Operaciones_unarias

Relacionales

Lógicos

Asignacion

AsignacionVector

DeclararFuncion

Parámetros

Statement

Callfuncion

Argumentos

ReturnExp

Transfer

Tolower

Toupper

Truncate

Round

Typeof

Tostring

Declararvector

Luego ya solo queda definir la gramática en si:

```
Init
    : Instrucciones EOF
;

Instrucciones
    : Instrucciones Instruccion
    | Instruccion
;

Instruccion
    : main
    | Print
    | Declaration
    | Asignacion
    | AsignacionVector
    | Callfuncion PUNTOCOMA
    | DeclararFuncion
    | Operaciones_unarias PUNTOCOMA
    | Declararvector
    | For
    | cif
    | While
    | Dowhile
```

```
    | ReturnExp
    | Transfer
;

main
    : MAIN Callfuncion PUNTOCOMA
;

Print
    : PRINT PARIZQ Expression PARDER PUNTOCOMA
;

Expression
    : Primitivo
    | Accvar
    | Accvector
    | Aritmetica
    | Relacionales
    | Operaciones_unarias
    | Logicos
    | Callfuncion
    | Ternario
    | Tolower
    | Toupper
    | Truncate
    | Round
    | Typeof
    | Tostring
```

```
;

Primitivo
    : INT
    | DOUBLE
    | BOOL
    | CHAR
    | CADENA
;

Declaration
    : Tipo IDENTIFICADOR PUNTOCOMA
    | Tipo IDENTIFICADOR IGUAL Expression PUNTOCOMA
;

Tipo
    : RES_INT
    | RES_DOUBLE
    | RES_BOOL
    | RES_CHAR
    | RES_STRING
;

Accvar
    : IDENTIFICADOR
;

Accvector
```

```
    : IDENTIFICADOR CORIZQ Expression CORDER
;

For
    : FOR PARIZQ Declaration Expression PUNTOCOMA Operaciones_unarias
PARDER Statement

While
    : WHILE PARIZQ Expression PARDER Statement
;

Dowhile
    : DO Statement WHILE PARIZQ Expression PARDER PUNTOCOMA
;

cif
    : IF PARIZQ Expression PARDER Statement celse
;

celse
    : ELSE Statement
    | ELSE cif
    |
;

Ternario
    : Expression KLENNE Expression DOSPUNTOS Expression
;
```

```
Aritmetica
    : Expression MAS Expression
    | Expression MENOS Expression
    | Expression POR Expression
    | Expression DIVISION Expression
    | Expression POTENCIA Expression
    | Expression MODULO Expression
    | MENOS Expression %prec UMENOS
    | PARIZQ Expression PARDER
;

Operaciones_unarias
    :IDENTIFICADOR INCREMENTO
    |IDENTIFICADOR DECREMENTO
;

Relacionales
    : Expression IGUALAR Expression
    | Expression NOIGUAL Expression
    | Expression MENOR Expression
    | Expression MENORIGUAL Expression
    | Expression MAYOR Expression
    | Expression MAYORIGUAL Expression
;

Logicos
    : Expression OR Expression
    | Expression AND Expression
```

```
    | NOT Expression
;

Asignacion
    : IDENTIFICADOR IGUAL Expression PUNTOCOMA
;

AsignacionVector
    : IDENTIFICADOR CORIZQ Expression CORDER IGUAL Expression PUNTOCOMA
;

DeclararFuncion
    : Tipo IDENTIFICADOR PARIZQ Parametros PARDER Statement
    | VOID IDENTIFICADOR PARIZQ Parametros PARDER Statement
    | Tipo IDENTIFICADOR PARIZQ PARDER Statement
    | VOID IDENTIFICADOR PARIZQ PARDER
;

Parametros
    : Parametros COMA Tipo IDENTIFICADOR
    | Tipo IDENTIFICADOR
;

Statement
    : LLAVEIZQ Instrucciones LLAVEDER
;

Callfuncion
```

```
    : IDENTIFICADOR PARIZQ PARDER
    | IDENTIFICADOR PARIZQ Argumentos PARDER
;

Argumentos
    : Argumentos COMA Expression
    | Expression

ReturnExp
    : RETURN Expression PUNTOCOMA
    | RETURN PUNTOCOMA

Transfer
    : BREAK PUNTOCOMA
    | CONTINUE PUNTOCOMA
;

//funciones nativas
Tolower
    : TOLOWER PARIZQ Expression PARDER
;

Toupper
    : TOUPPER PARIZQ Expression PARDER
;

Truncate
    : TRUNCATE PARIZQ Expression PARDER
```

```
;

Round
    : ROUND PARIZQ Expression PARDER
;

Typeof
    : TYPEOF PARIZQ Expression PARDER
;

Tostring
    : TOSTRING PARIZQ Expression PARDER
;

Declararvector
    : Tipo CORIZQ CORDER IDENTIFICADOR IGUAL NEW Tipo CORIZQ Expression
CORDER PUNTOCOMA
    | Tipo CORIZQ CORDER IDENTIFICADOR IGUAL LLAVEIZQ Argumentos
LLAVEDER PUNTOCOMA
 ;
```