ARCHIVO DE GRAMÁTICAS DE “SYSCOMPILER”

```
ini
    :general EOF{
        return $1;
    }
;

general
        :general cuerpo
        |cuerpo
;

cuerpo
    :COMENTARIO
    |funcionMetodo
    |start with ID ( ) ;
    |start with ID ( expresion ) ;
    |decl
;

funcionMetodo
        :tipo ID ( parametros ) { statement }
        |tipo ID ( ) { statement }

        |DynamicList < tipo > ID ( parametros ) { statement }
        |DynamicList < tipo > ID ( ) { statement }

        |tipo ID [ ] ( parametros ) { statement }
        |tipo ID [ ] ( ) { statement }

        |void ID ( parametros ) { statement }
        |void ID ( ) { statement }
;

parametros
        :parametros , parametros
        |tipo ID
        |DynamicList < tipo > ID
        |tipo ID [ ]
;
```

```
cuerpoFunciones
            :cuerpoFunciones declaraciones
            |declaraciones
;

declaraciones
          :sentencias
          |WriteLine ( listaExpresiones ) ;
          |break ;
          |continue ;
          |return expresion ;
          |return ;
          |COMENTARIO
          |decl
;

decl
   :variable
   |vectores
   |listas
   |error ;
;

sentencias
          :if
          |switch
          |while
          |for
          |dowhile
;

dowhile
       :do { statement } while ( expresion ) ;
;

for
   :for ( declaracionFor ; expresion ; declaracionFor ) { statement }
   |for ( declaracionFor ; expresion ; ID + + ) { statement }
   |for ( declaracionFor ; expresion ; ID - - ) { statement }
```

```
;

declaracionFor
        :tipo ID = expresion
        |ID = expresion
        |tipo ID = casteos
        |ID = casteos
;

switch
    :switch ( expresion ) { casos }
;

if
    :if ( expresion ) { statement } else
;

else
    :else { statement }
    |else if
    |
;

while
    :while ( expresion ) { statement }
;

statement
        :cuerpoFunciones
        |
;

casos
    :casos caso
    |caso
;

caso
    :case expresion : statement
    |default : statement
```

```
;

variable
        :tipo identificadores ;
        |tipo identificadores = expresion ;
        |identificadores = expresion ;
        |tipo identificadores = casteos
        |identificadores = casteos
        |ID + + ;
        |ID - - ;
        |ID ( listaExpresiones ) ;
        |ID ( ) ;
;

vectores
        :tipo ID [ ] = new tipo [ expresion ] ;
        |tipo ID [ ] = { listaVectores } ;
        |ID [ expresion ] = expresion ;
        |ID [ ] = new tipo [ expresion ] ;
        |ID [ ] = { listaVectores } ;
;

listas
    :DynamicList < tipo > ID = new DynamicList < tipo > ;
    |append ( ID , expresion ) ;
    |setValue ( ID , aritmeticos , expresion ) ;
    |DynamicList < tipo > ID = toCharArray ( expresion )  ;
;

listaVectores
        :listaVectores , listaVectores
        |expresion
;

listaExpresiones
            :listaExpresiones , expresion
            |expresion
;

aritmeticos
```

```
        :ID
        |aritmeticos + aritmeticos
        |aritmeticos - aritmeticos
        |aritmeticos * aritmeticos
        |aritmeticos / aritmeticos
        |aritmeticos % aritmeticos
        |aritmeticos ^ aritmeticos
        |ENTERO
        |DECIMAL
        |ID [ expresion ]
        |getValue ( ID , expresion )
        |( aritmeticos )
;

casteos
        :( tipo ) expresion ;
;

identificadores
            : identificadores , identificadores
            | ID
;

tipo
    :int
    |boolean
    |double
    |char
    |string
;

expresion
    :- expresion %prec UMENOS
    |! expresion
    |expresion && expresion
    |expresion || expresion
    |expresion + expresion
    |expresion - expresion
    |expresion * expresion
    |expresion / expresion
```

```
    |expresion % expresion
    |expresion ^ expresion
    |( expresion )
    |expresion == expresion
    |expresion != expresion
    |expresion >= expresion
    |expresion <= expresion
    |expresion > expresion
    |expresion < expresion
    |expresion + +
    |expresion - -
    |ID

    |ID [ expresion ]
    |getValue ( ID , expresion )

    |ID ( listaExpresiones )
    |ID ( )

    |toLower ( expresion )
    |toUpper ( expresion )
    |length ( expresion )
    |truncate ( expresion )
    |round ( expresion )
    |typeOf ( expresion )
    |toString ( expresion )
    |toCharArray ( expresion )

    |ENTERO
    |DECIMAL
    |true
    |false
    |CARACTER
    |FRASE
    |expresion ? expresion : expresion
;
```

DESCRIPCIÓN DE LAS PRODUCCIONES:

general:

Contiene todo el contenido de las producciones.

cuerpo:

Contiene el cuerpo del analizador.

funcionMetodo:

Contiene las formas de declarar las funciones y métodos con o sin parámetros.

parametros:

Contiene la forma de declarar los parámetros para las funciones y métodos.

cuerpoFunciones:

Contiene la lista de las declaraciones del cuerpo de las funciones y métodos.

declaraciones:

Contiene la forma en que se pueden declarar las sentencias, los comentarios y las distintas cosas que subirá a la lista del cuerpo de las funciones.

decl:

Contiene la forma en que se pueden declarar las variables, vectores, listas y la recuperación de errores.

sentencias:

Contiene la forma en que se pueden declarar todas las sentencias.

dowhile:

Forma de declarar el do-while.

for:

Forma de declarar el for.

declaracionFor:

Forma de declarar lo que va dentro del for.

switch:
Forma de declarar el switch.

if:
Forma de declarar el if.

else:
Forma de declarar el else.

while:
Forma de declarar el while.

statement:
Forma de declarar el cuerpo que va dentro de las sentencias y funciones.

casos:
Forma de declarar la lista de casos que van dentro del switch.

caso:
Forma de declarar el caso que subirá a la lista de casos.

variable:
Forma de declarar las variables.

vectores:
Forma de declarar los vectores.

listas:
Forma de declarar las listas.

listaVectores:
Forma de declarar el do-while.

listaExpresiones:
Forma de declarar la lista de expresiones que va al declarar un vector.

aritmeticos:
Forma de declarar un valor entero para una posición del vector.

casteos:

Forma de declarar los casteos posibles dentro del programa.

identificadores:

Forma de declarar la lista de identificadores.

tipo:

tipos de datos permitidos en el programa.

expresion:

distintas expresiones permitidas dentro del lenguaje, siendo expresiones lógicas, relacionales y aritméticas.

EXPRESIONES REGULARES:

```
("//".*\r\n)|("//".*\n)|("//".*\r)          return "COMENTARIO";
"/*"/"*([^*/]|[^*]"/"|"*"[^/])*"*"*"*/"     return "COMENTARIO";
[\"]([^\"\n]|(\\\"))*[\"]                    return "FRASE";
[\']([^\"\n]|(\\\"))[\']                     return "CARACTER";
```