Expresiones regulares

Nombre	Expresión R
num	[0-9]+
id	[a-zñA-ZÑ_][a-zñA-ZÑ0-9_]*
escapechar	[\'\"\\nrt]
escape	\\{escapechar}
aceptación	[^\"\\]
cadena	(\" ({escape}|{aceptacion})*\")
escapechar2	[\'\"\\nrt]
escape2	\\{escapechar2}
aceptacion2	[^\'\\]
caracter	(\' ({escape2}|{aceptacion2})\')

Precedencia de operadores

Operador	Descripcion	Asociatividad
?		Derecha
(	Signo de agrupación	Derecha
||	OR	Izquierda
&	AND	Izquierda
!	NOT	Derecha
== != < <= > >=	Igualacion Diferenciacion Menor que Menor o igual que Mayor que Mayor o igual que	Izquierda
+ -	Suma Resta	Izquierda
* /	Multiplicacion Division	Izquierda
^	Potencia	Izquierda
%	Modulo	Izquierda
-	Negacion Unaria	Derecha

Terminales

Nombre	Explicación
TRUE	Define un valor verdadero
FALSE	Define un valor falso
INT	Define un tipo de valor entero
STRING	Define un tipo de valor cadena
DOUBLE	Define un tipo de valor doublé
CHAR	Define un tipo de valor char

BOOLEAN	Define un tipo de valor boolean
VOID	Define un tipo de valor void
WRITELINE	Define la función para imprimir en consola
TOLOWER	Define la función para convertir en minúsculas una cadena
TOUPPER	Define la función para convertir en mayúsculas una cadena
TRUNCATE	Define la función para eliminar los decimales de un número
ROUND	Define la función para redondear los números decimales
TYPEOF	Define la función para retornar una cadena con el tipo del dato
TOSTRING	Define la función para convertir de tipo numérico o booleano a texto
IF	Símbolo para el inicio de una sentencia condicional
ELSE	Símbolo para indicar las instrucciones que deben ejecutarse si la instrucción de arriba no se cumple
WHILE	Símbolo para el inicio del bucle, que ejecuta el bloque de instrucciones mientras no se cumpla la condición indicada
BREAK	
SWITCH	Símbolo para el inicio de una sentencia de control.
CASE	Símbolo para definir un caso de una sentencia Switch
DO	Símbolo para el inicio del bucle, que ejecuta el bloque de instrucciones mientras no se cumpla la condición que se colocó al final
DEFAULT	Define un caso predeterminado
FOR	Símbolo para el inicio de una sentencia cíclica
DYNAMICLIST	Símbolo para definir una lista
NEW	Símbolo para deinir un nuevo objeto
APPEND	Símbolo para agregar un dato a una lista
SETVALUE	Símbolo para obtener un valor
GETVALUE	Símbolo para guardar un valor
CONTINUE	Símbolo para saltarse una instrucción en un bucle
RETURN	Símbolo para retornar un valor en una función o instrucción
START	Símbolo para indicar el inicio del programa
WITH	Símbolo para indicar el inicio del programa
INCRE	Define dos más en la gramática
DECRE	Define dos menos en la gramática
IGUALIGUAL	Define dos igual en la gramática
DIFERENTE	Define un signo de exclamación y un igual en la gramática
PARA	Define un paréntesis que abre (.
PARC	Define un paréntesis que cierra).
CORA	Define un corchete que abre [.
CORC	Define un corchete que cierra].
LLAVA	Define una llave que abre {.
LLAVC	Define una llave que cierra }.
COMA	Define una coma ,
PYC	Define un punto y coma ;

IGUAL	Define un igual =
INTERROGACION	Define un signo de interrogación ¿
DOSPUNTOS	Define dos puntos :
MAS	Define un más +
MENOS	Define un menos -
MULTI	Define un asterisco *
DIV	Define una división /
POT	Define una potencia ^
NOT	Define una negación -
MOD	Define un modulo %
MENORIGUAL	Define un menor igual <=
MENORQUE	Define un menor que <
MAYORIGUAL	Define un mayor igual >=
MAYORQUE	Define un mayor que >

No terminales

Inicio	Define el inicio de la gramática
Instrucciones	Este recibe una lista de instrucciones
Instrucción	Recibe las instrucciones
Declaración	Recibe una lista de tipos y declaraciones
Tipo	Recibe los tipos
Decl_vectores	Recibe una lista de valores
Lista_valores	Recibe los valores
Modi_vector	Recibe la expresión a modificar
Decl_list_din	Recibe una
Agregar_lista	Recibe los valore a ingresar
Modi_lista	Recibe los valores a modificar
Lista_id	Recibe una lista de ids
Writeline	Recibe la expresión a imprimir
Asignacion	Recibe el id y valor a cambiar
Sent_if	Recibe una lista de instrucciones
Set_switch	Recibe una lista de casos
List_case	Recibe una lista de casos
Caso	Recibe los casos
Sent_while	Recibe una lista de instrucciones
Sent_for	Recibe una lista de instrucciones
Dec_asignacion_for	Recibe una asignación de un token
Sent_do_while	Recibe una lista de instrucciones
Funciones	Recibe las instrucciones
Lista_parametros	Recibe una lista de parámetros
Llamada	Recibe la lista de valores o solo el id
Startwith	Recibe la función de cual iniciar
e	Recibe una lista de expresiones

Inicio de la gramática

```
inicio: instrucciones;
```

Instrucciones:

```
instrucciones : instrucciones instruccion
              | instruccion
              ;

instruccion : declaracion
            | asignacion
            | ID ++ ';'
            | ID -- ';'
            | declaracion_vectores
            | modificacion_vector
            | declaracion_lista
            | agregar_lista
            | modificacion_lista
            | sent_if
            | sent_switch
            | sent_while
            | sent_for
            | sent_do_while
            | 'break' ';'
            | 'continue' ';'
            | 'return' ';'
            | return e ;
            | llamada ;
            | funciones
            | 'writeline' ( e ) ;
            ;
```

Declaración:

```
declaracion : tipo lista_id';'
            | tipo lista_id = e ';'
            ;

lista_id : lista_ids , ID
         | ID
         ;

tipo : INT
     | DOUBLE
     | STRING
     | CHAR
     | BOOLEAN
     | VOID
     ;
```

Asignación:

```
asignacion : ID = e ;
           ;
```

Startwith:

```
startwith : 'start' 'with' ID ();
          | 'start' 'with' ID (lista_vals);
```

Vectores:

```
declaracion_vectores : tipo ID [] = 'new' tipo [e] ';'
                     | tipo ID [] = { lista_valores }
                     | tipo ID [] = e
                     ;

modificacion_vector : ID [e] = e';'
                    ;

lista_valores : lista_valores , e
              | e
              ;
```

Expresiones:

```
e
    : e MAS e
    | e MENOS e
    | e MULTI e
    | e DIV e
    | e POT e
    | e MOD e
    | e MAYORIGUAL e
    | e MAYORQUE e
    | e MENORIGUAL e
    | e MENORQUE e
    | e IGUALIGUAL e
    | e DIFERENTE e
    | e AND e
    | e OR e
    | e NOT
    | MENOS e %prec UMINUS
    | PARA e PARC
    | DECIMAL
    | ENTERO
    | ID
    | CADENA
    | CARACTER
    | TRUE
    | FALSE
    | e ? e : e
    | ( tipo )  e
    | ID ++
    | ID --
    | ID [ e ]  // obtener valor del vector
    | 'getvalue' ( e , e) // obtener valor de la lista
    | llamada
    | 'tolower' ( e )
    | 'toupper' ( e )
    | 'length' ( e )
    | 'truncate' ( e )
    | 'round' ( e )
    | 'typeof' ( e )
    | 'tostring' ( e )
    | 'tochararray' ( e )
    | startwith
    ;
```

Listas dinámicas:

```
declaracion_lista : 'dinamiclist' '<' tipo '>' ID  = 'new' 'dinamiclist' '<' tipo '>' ';'
                  | 'dinamiclist' '<' tipo '>' ID = e ;
                  ;

agregar_lista : 'append' ( e ',' e) ';'
              ;


modificacion_lista : 'setvalue' ( e ,e , e)';'
                   ;
```

Sentencia if

```
sent_if : 'if' ( e ) { instrucciones}
        | 'if' ( e ) { instrucciones } 'else' {instrucciones}
        | 'if' ( e ) { instrucciones } 'else' sent_if
        ;
```

Sentencia Switch:

```
sent_switch : 'switch' ( e ) {lista_case }
            | 'switch' ( e ) {lista_case default }
            | 'switch' ( e ) {default}
            ;

lista_case : lista_case caso
           | caso
           ;

caso : 'case' e : instrucciones
     ;

 default : 'default' : instrucciones
```

Sentencia while:

```
sent_while : 'while' ( e ) { instrucciones }
           ;
```

Sentencia for:

```
sent_for : 'for' (decla_asig_for ';' e ';' actualizacion_for ';') { instrucciones}
         ;


decla_asig_for : tipo ID = e
               | ID = e
               ;

actualizacion_for : ID '++'
                  | ID '--'
                  | id '=' e
                  ;
```

Sentencia do-while:

```
sent_do_while : 'do' { instrucciones } 'while' '('e')' ';'
              ;
```

Funciones y métodos:

```
funciones : tipo ID (lista_params ) { instrucciones}
          | tipo ID () { instrucciones}
          ;

lista_params : lista_params , tipo ID
             | tipo ID
             ;
```

Llamada:

```
llamada : ID ( lista_valores )
        | ID ()
        ;
```