

Análisis de gramática

TERMINALES

NOMBRE	SIMBOLO	NOMBRE	SIMBOLO
Rint	int	Rvoid	void
Rdouble	double	Rprint	print
Rboolean	boolean	Rtrue	true
Rchar	char	Rfalse	false
Rstring	string	Rmain	main
Rif	if	Rnew	new
Relse	else	Radd	add
Rswitch	switch	RtoLower	toLower
Rcase	case	RtoUpper	toUpper
Rdefault	default	Rlength	length
Rbreak	break	Rtruncate	truncate
Rreturn	return	Rround	round
Rcontinue	continue	Rtypeof	typeof
Rfor	for	RtoString	toString
Rwhile	while	RtoCharArray	toCharArray
Rdo	do	decimal	[0-9]+("."[0-9]+)
entero	[0-9]+	punto	"."
igualigual	"=="	ptcoma	","
diferente	"!="	dospuntos	":"
menorigual	"<="	inter	"?"
menor	"<"	and	"&&"

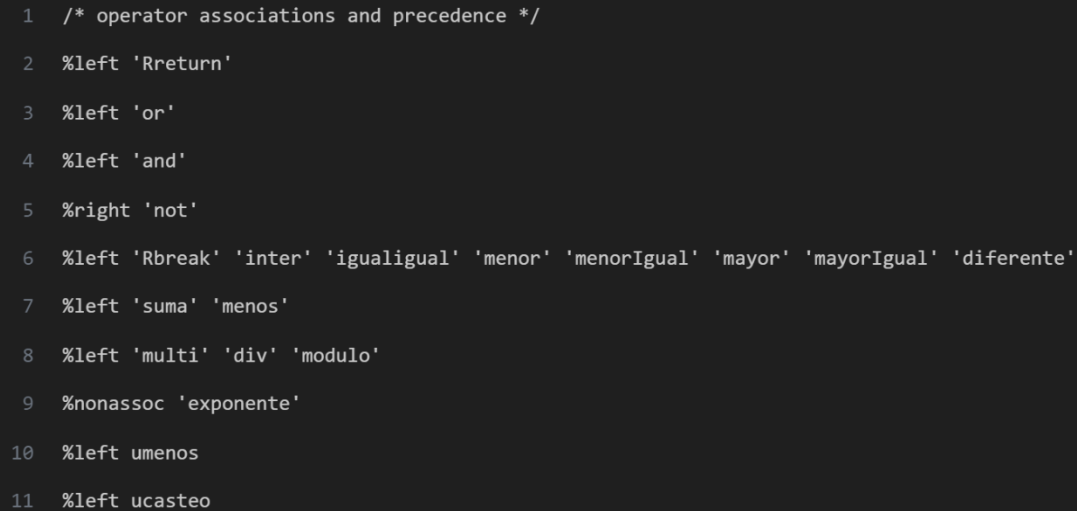
NOMBRE	SIMBOLO	NOMBRE	SIMBOLO
mayorIgual	">="	llaveA	"{"
igual	"="	llaveC	"}"
mayor	">"	multi	"*"
coma	","	div	"/"
menosmenos	"--"	not	"!"
masmas	"++"	modulo	"%"
menos	"_"	parA	"("
suma	"+"	parC	")"
exponente	"^"	corchA	"["
corchC	"]"	Rlist	list

NO TERMINALES

NOMBRE	NOMBRE	NOMBRE	NOMBRE
INICIO	PARAMETROS	TIPO	FOR
OPCIONESCUEPO	MAIN	INSTRUCCIONES	WHILE
CUEPO	PARAMETROS_LLAMADA	INSTRUCCION	DOWHILE
METODOS	DEC_VAR	PRINT	INCREMENTOYDECREMENTO
LIST_PARAMETROS	ASIG_VAR	IF	LISTA
SWITCH	DEC_ESTRUCT	MODVECTOR	ADD_LIST

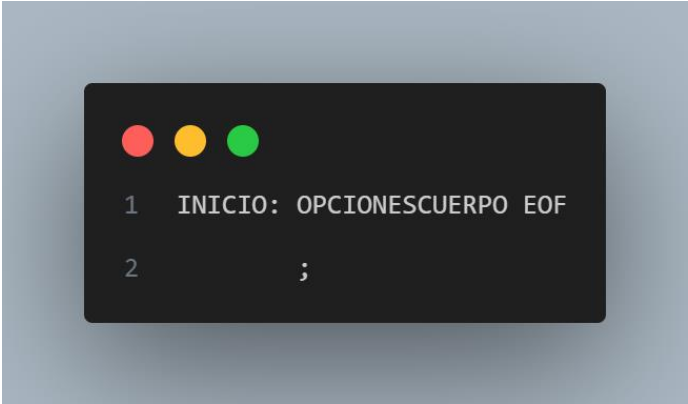
En la sección léxica se definen todos los símbolos que se utilizan en las instrucciones. Cuando se habla de una palabra reservada, se utiliza `Rpalabrarreservada` para utilizarla en las gramáticas y producciones.

En la siguiente sección se definen las precedencias para que la gramática evite ambigüedades y busque primero lo que se decide en esta sección.



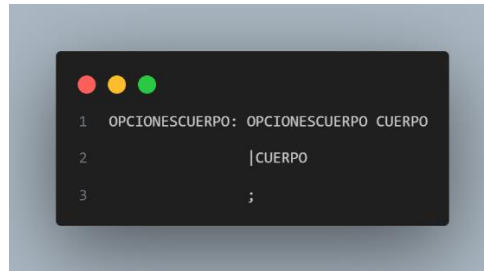
```
1  /* operator associations and precedence */
2  %left 'Rreturn'
3  %left 'or'
4  %left 'and'
5  %right 'not'
6  %left 'Rbreak' 'inter' 'igualigual' 'menor' 'menorIgual' 'mayor' 'mayorIgual' 'diferente'
7  %left 'suma' 'menos'
8  %left 'multi' 'div' 'modulo'
9  %nonassoc 'exponente'
10 %left umenos
11 %left ucasteo
```

El inicio de la producción se define con `%start`, en este caso es el no terminal `INICIO`. De ese no terminal parten todas las producciones a trabajar. Donde se evalúa



```
1  INICIO: OPCIONESCUERPO EOF
2  ;
```

De las opciones cuerpo parte el no terminal cuerpo, el cual define una recursividad por la izquierda para analizar todo lo que pueda venir.

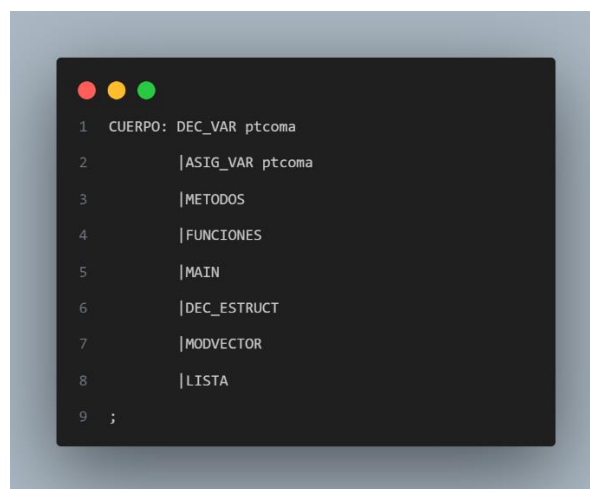


```

1 OPCIONESCUERPO: OPCIONESCUERPO CUERPO
2               | CUERPO
3               ;

```

Del cuerpo parten los no terminales que pueden estar en el ámbito global, es decir que pueden estar fuera de cualquier método o función. Estas son declaraciones de variables, asignaciones de variables, métodos, funciones, main, declaración estructura, modificar vector, lista.

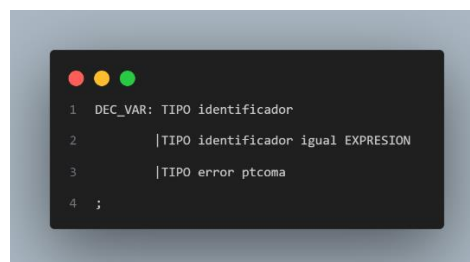


```

1 CUERPO: DEC_VAR ptcoma
2       | ASIG_VAR ptcoma
3       | METODOS
4       | FUNCIONES
5       | MAIN
6       | DEC_ESTRUCT
7       | MODVECTOR
8       | LISTA
9       ;

```

- Declarar variable: debe tener un tipo, identificador y puede o no estar igualado a una expresión.

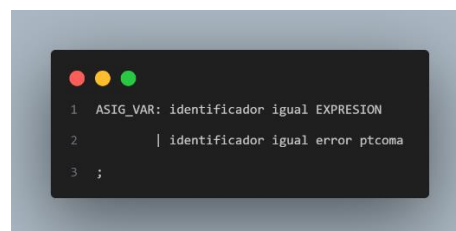


```

1 DEC_VAR: TIPO identificador
2        | TIPO identificador igual EXPRESION
3        | TIPO error ptcoma
4        ;

```

- Asignar variable: debe tener el identificador y estar igualada a una expresión.



```

1 ASIG_VAR: identificador igual EXPRESION
2         | identificador igual error ptcoma
3         ;

```

- Métodos: deben tener void, identificador con sus parámetros o no dentro de paréntesis. Y dentro de llaves las instrucciones a realizar.

```

1  METODOS: Rvoid identificador parA parC llaveA INSTRUCCIONES llaveC
2          | Rvoid identificador parA LIST_PARAMETROS parC llaveA INSTRUCCIONES llaveC
3  ;

```

- Funciones: Deben tener tipo, identificador, sus parámetros entre paréntesis y puede que sea una función que solo devuelva una expresión con el return, o bien puede tener instrucciones y devolver una expresión con el return.

```

1  FUNCIONES: TIPO identificador parA LIST_PARAMETROS parC llaveA INSTRUCCIONES Rreturn EXPRESION ptcoma llaveC
2          | TIPO identificador parA LIST_PARAMETROS parC llaveA Rreturn EXPRESION ptcoma llaveC
3          | error llaveC
4  ;

```

- Main: donde debe de venir la palabra main, identificador de método y puede venir o no con parámetros en la llamada.

```

1  MAIN: Rmain identificador parA parC ptcoma
2          |Rmain identificador parA PARAMETROS_LLAMADA parC ptcoma
3
4  ;

```

- Declarar estructura: para esto se necesita el tipo, corchetes, identificador y que sea igualdo e instanciado un nuevo objeto. Este se puede instanciar con una expresión o con una lista de valores.

```
1 DEC_ESTRUCT: TIPO corchA corchC identificador igual Rnew TIPO corchA EXPRESION corchC ptcoma
2           | TIPO corchA corchC identificador igual llaveA LISTA_VALORES llaveC ptcoma
3 ;
```

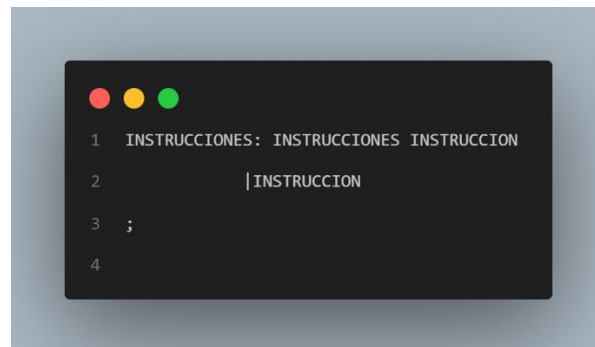
- Modificar vector: se coloca el identificador, entre corchetes la posición representada por una expresión y a lo que se iguala.

```
1 MODVECTOR: identificador corchA EXPRESION corchC igual EXPRESION ptcoma
2 ;
```

- Lista: se utiliza la palabra list, entre signos se coloca el tipo y se iguala a nueva nueva lista instanciada

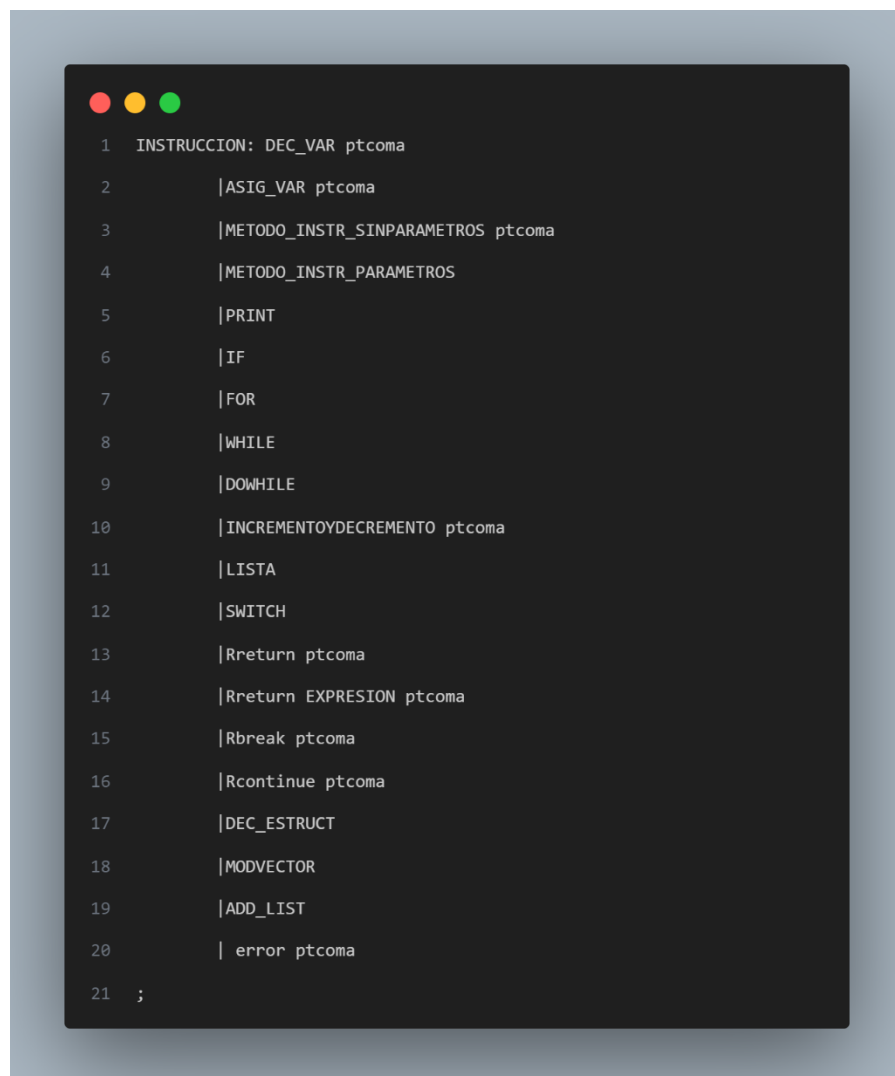
```
1 LISTA: Rlist menor TIPO mayor identificador igual Rnew Rlist menor TIPO mayor ptcoma
2       | Rlist error ptcoma
3 ;
```

Cuando se hace la llamada a los métodos, es obvio que van a tener más instrucciones adentro de la estructura. Es por eso por lo que se realiza una producción con recursividad por la izquierda con las instrucciones.



```
1 INSTRUCCIONES: INSTRUCCIONES INSTRUCCION
2         |INSTRUCCION
3 ;
4
```

- Instrucción: se define cada instrucción que puede venir con no terminales y terminales.



```
1 INSTRUCCION: DEC_VAR ptcoma
2         |ASIG_VAR ptcoma
3         |METODO_INSTR_SINPARAMETROS ptcoma
4         |METODO_INSTR_PARAMETROS
5         |PRINT
6         |IF
7         |FOR
8         |WHILE
9         |DOWHILE
10        |INCREMENTOYDECREMENTO ptcoma
11        |LISTA
12        |SWITCH
13        |Rreturn ptcoma
14        |Rreturn EXPRESION ptcoma
15        |Rbreak ptcoma
16        |Rcontinue ptcoma
17        |DEC_ESTRUCT
18        |MODVECTOR
19        |ADD_LIST
20        | error ptcoma
21 ;
```

Y se realiza el mismo proceso las estructuras definidas en la instrucción.