

Especificaciones de Gramáticas, Análisis Sintáctico y Notación BNF

(Forma Normal de Backüs)

(En construcción)

Notas Materia LENGUAJES DE PROGRAMACION
María de Guadalupe Cota Ortiz

Gramática Libre de Contenido

$\langle S \rangle ::= (\langle S \rangle) \langle S \rangle$

$\langle S \rangle ::= a$

Gramática Libre de Contenido

$\langle S \rangle ::= (\langle S \rangle) \langle S \rangle$

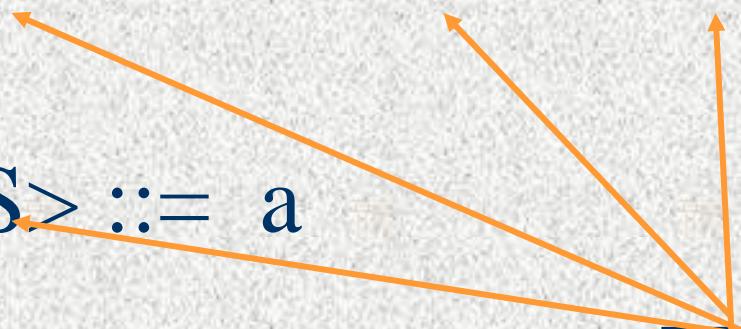
$\langle S \rangle ::= a$

Terminales

Gramática Libre de Contenido

$$<S> ::= (<S>) <S>$$
$$<S> ::= a$$

No-terminales



Gramática Libre de Contex

$\langle S \rangle ::= (\langle S \rangle) \langle S \rangle$

$\langle S \rangle ::= a$

Símbolo inicial: $\langle S \rangle$

Gramática Libre de Contenido

$$\langle S \rangle ::= (\langle S \rangle) \langle S \rangle$$
$$\langle S \rangle ::= a$$

Regla o Producción

Gramática Libre de Contenido

$\langle \text{expr} \rangle ::= \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle) | \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= - \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= \langle \text{num} \rangle$

$\langle \text{op} \rangle ::= +$

Terminales:

$\langle \text{op} \rangle ::= *$

(,),-,+,*,**0,1,2,3,4,5,6,7,8,9**

$\langle \text{num} \rangle ::= \langle \text{dig} \rangle | \langle \text{num} \rangle$

$\langle \text{dig} \rangle ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$

Gramática Libre de Contenido

$\langle \text{expr} \rangle ::= \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle) | \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= - \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= \langle \text{num} \rangle$

$\langle \text{op} \rangle ::= +$

$\langle \text{op} \rangle ::= *$

No Terminales:

expr, op, num, dig

$\langle \text{num} \rangle ::= \langle \text{dig} \rangle | \langle \text{num} \rangle$

$\langle \text{dig} \rangle ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$

Gramática Libre de Contenido

$\langle \text{expr} \rangle ::= \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle) | \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= - \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= \langle \text{num} \rangle$

$\langle \text{op} \rangle ::= +$

$\langle \text{op} \rangle ::= *$

Símbolo inicial:

expr

$\langle \text{num} \rangle ::= \langle \text{dig} \rangle | \langle \text{num} \rangle$

$\langle \text{dig} \rangle ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$

Gramática Libre de Contenido

$\langle \text{expr} \rangle ::= \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle) | \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= - \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= \langle \text{num} \rangle$

$\langle \text{op} \rangle ::= +$

$\langle \text{op} \rangle ::= *$

$\langle \text{num} \rangle ::= \langle \text{dig} \rangle | \langle \text{num} \rangle$

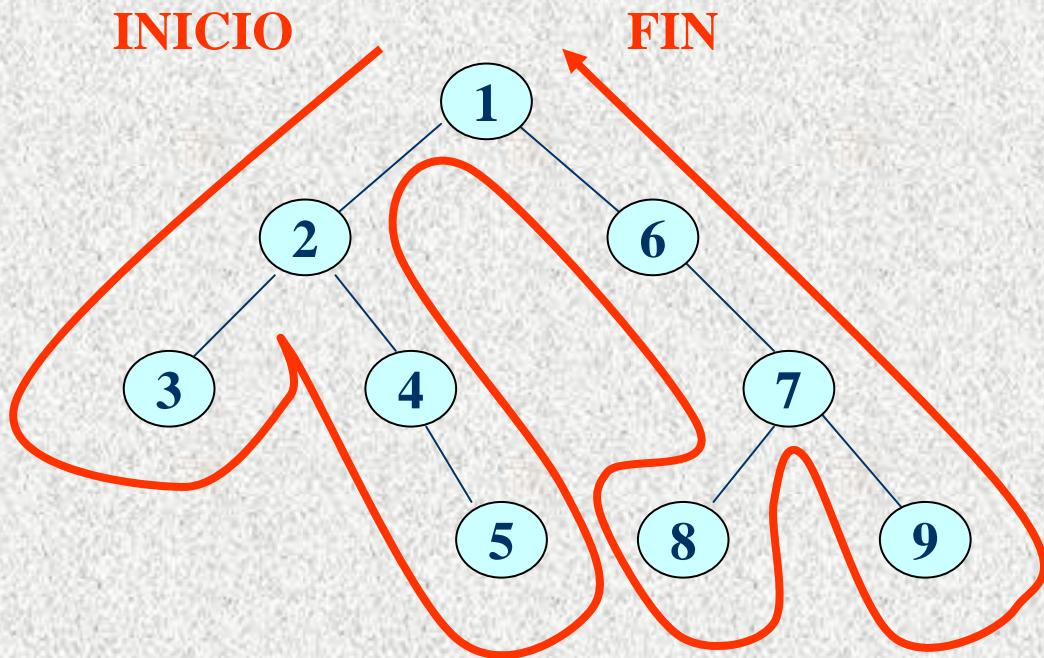
$\langle \text{dig} \rangle ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$

Producciones

Recordatorio

(Recorridos)

Recorrido Pre-Orden (Derivación izquierda)

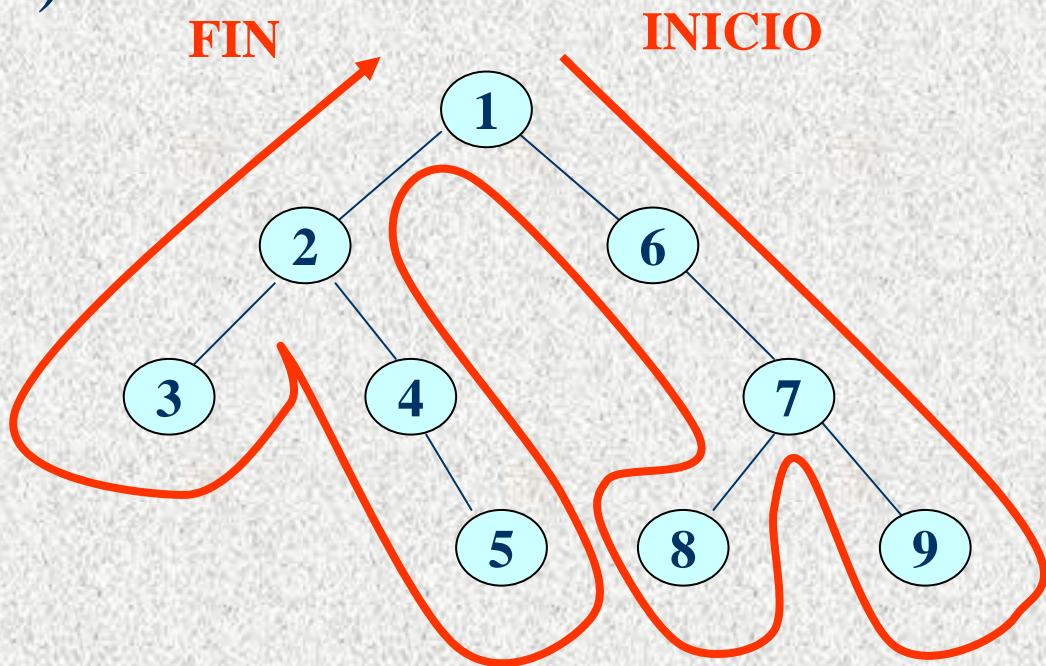


Recorrido: 1,2,3,4,5,6,7,8,9

Recordatorio

(Recorridos)

Recorrido POS-Orden-Inverso (Derivación derecha)



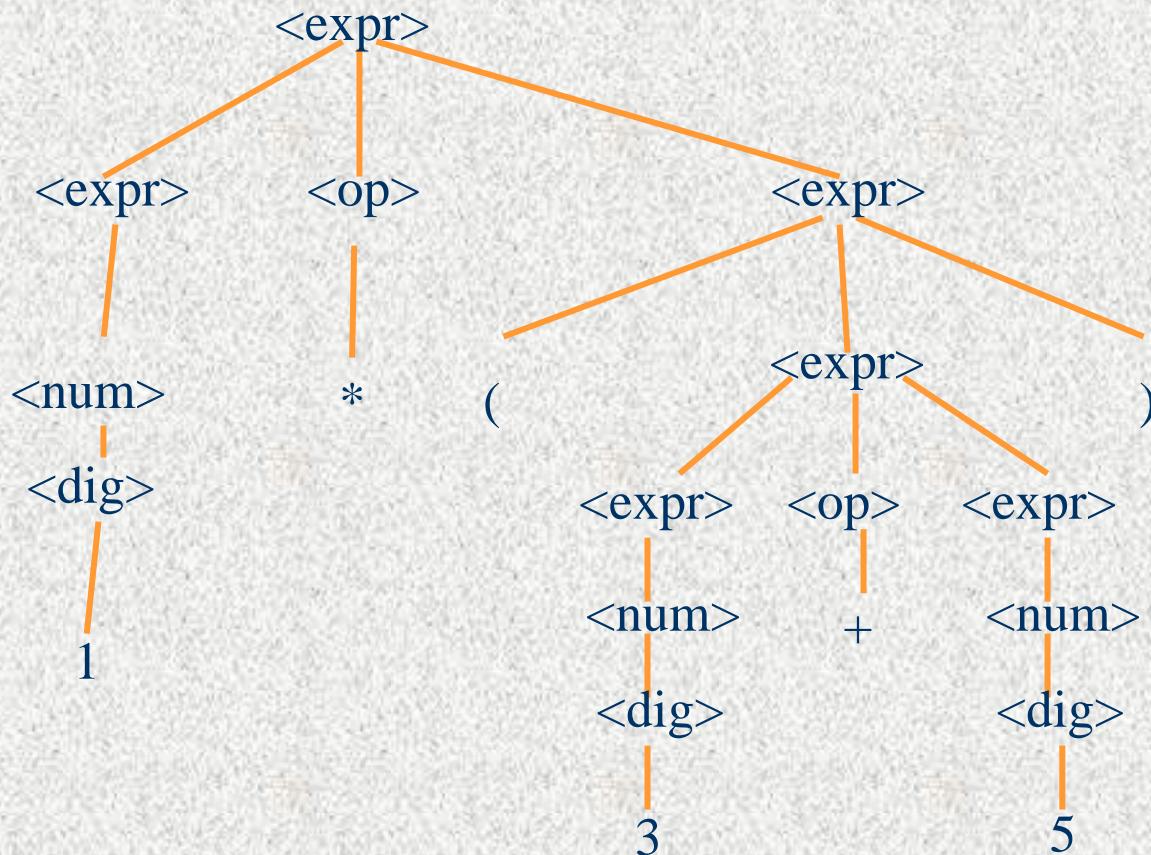
Recorrido: 1,6,7,9,8,2,4,5,3

Arbol de Parseo

Derivación por la izquierda

$\langle \text{expr} \rangle ::= \langle \text{num} \rangle$

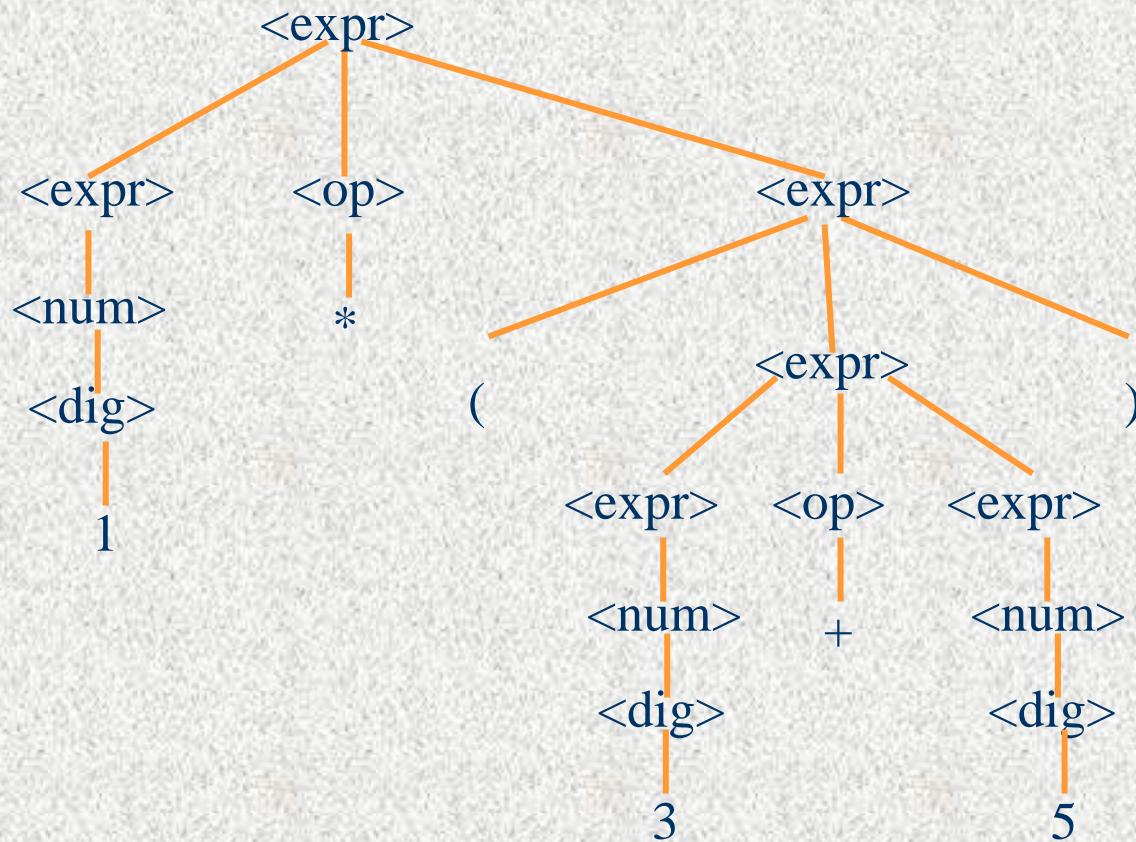
Cadena: 1 '*' '(' 3 '+' 5 ')'



Arbol de Parseo Derivación por la derecha

Producción: $\langle \text{expr} \rangle \Rightarrow \text{num}$

Cadena: 1 '*' '(' 3 '+' 5 ')'



Parser ‘Top-down’ (LL)

⇒ <expr>
⇒ <expr> <op> <expr>
⇒ <num> <op> <expr>
⇒ <dig> <op> <expr>
⇒ ‘1’ <op> <expr>
⇒ ‘1’ ‘*’ <expr>
⇒ ‘1’ ‘*’ ‘(‘ <expr> ‘)’
⇒ ‘1’ ‘*’ ‘(‘ <expr> <op> <expr> ‘)’
⇒ ‘1’ ‘*’ ‘(‘ <num> <op> <expr> ‘)’
⇒ ‘1’ ‘*’ ‘(‘ <dig> <op> <expr> ‘)’
⇒ ‘1’ ‘*’ ‘(‘ ‘3’ <op> <expr> ‘)’
⇒ ‘1’ ‘*’ ‘(‘ ‘3’ ‘+’ <expr> ‘)’
⇒ ‘1’ ‘*’ ‘(‘ ‘3’ ‘+’ <num> ‘)’
⇒ ‘1’ ‘*’ ‘(‘ ‘3’ ‘+’ <dig> ‘)’
⇒ ‘1’ ‘*’ ‘(‘ ‘3’ ‘+’ <5> ‘)’

Parser ‘Bottom-Up’ (LR)

→ $\langle \text{expr} \rangle$
→ $\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$
→ $\langle \text{expr} \rangle \langle \text{op} \rangle ' (' \langle \text{expr} \rangle ')'$
→ $\langle \text{expr} \rangle \langle \text{op} \rangle ' (' \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle ')'$
→ $\langle \text{expr} \rangle \langle \text{op} \rangle ' (' \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{num} \rangle ')'$
→ $\langle \text{expr} \rangle \langle \text{op} \rangle ' (' \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{dig} \rangle ')'$
→ $\langle \text{expr} \rangle \langle \text{op} \rangle ' (' \langle \text{expr} \rangle '+' \langle \text{dig} \rangle ')'$
→ $\langle \text{expr} \rangle \langle \text{op} \rangle ' (' \langle \text{num} \rangle '+' \langle \text{dig} \rangle ')'$
→ $\langle \text{expr} \rangle \langle \text{op} \rangle ' (' \langle \text{dig} \rangle '+' \langle \text{dig} \rangle ')'$
→ $\langle \text{expr} \rangle '*' ' (' \langle \text{dig} \rangle '+' \langle \text{dig} \rangle ')'$
→ $\langle \text{num} \rangle '*' ' (' \langle \text{dig} \rangle '+' \langle \text{dig} \rangle ')'$
→ $\langle \text{dig} \rangle '*' ' (' \langle \text{dig} \rangle '+' \langle \text{dig} \rangle ')'$

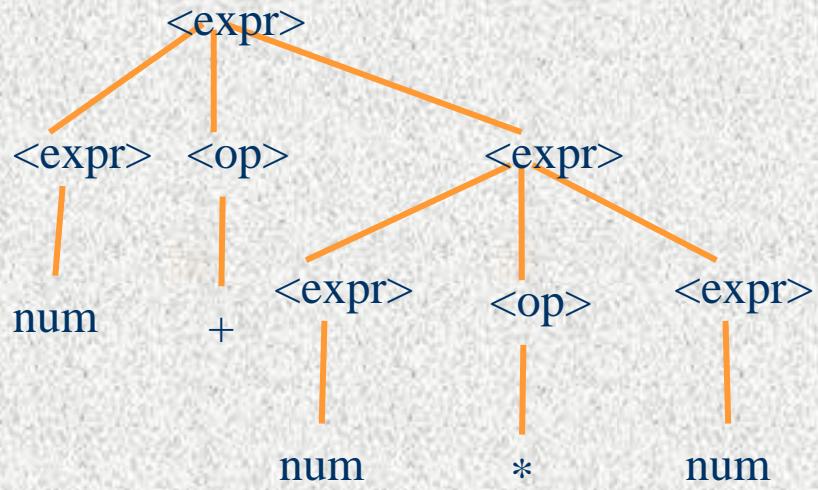
(Se invierte el orden de evaluación)

Parser ‘Bottom-Up’ (LR)

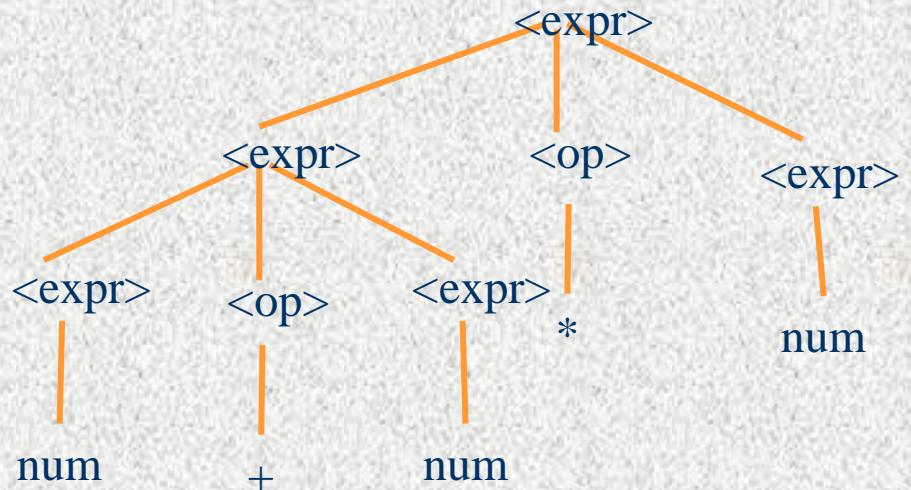
⇒ $\langle \text{dig} \rangle^* ('(\langle \text{dig} \rangle^+ \langle \text{dig} \rangle^*)'$
⇒ $\langle \text{num} \rangle^* ('(\langle \text{dig} \rangle^+ \langle \text{dig} \rangle^*)'$
⇒ $\langle \text{expr} \rangle^* ('(\langle \text{dig} \rangle^+ \langle \text{dig} \rangle^*)'$
⇒ $\langle \text{expr} \rangle \langle \text{op} \rangle ('(\langle \text{dig} \rangle^+ \langle \text{dig} \rangle^*)'$
⇒ $\langle \text{expr} \rangle \langle \text{op} \rangle ('(\langle \text{num} \rangle^+ \langle \text{dig} \rangle^*)'$
⇒ $\langle \text{expr} \rangle \langle \text{op} \rangle ('(\langle \text{expr} \rangle^+ \langle \text{dig} \rangle^*)''$
⇒ $\langle \text{expr} \rangle \langle \text{op} \rangle ('(\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{dig} \rangle^*)'$
⇒ $\langle \text{expr} \rangle \langle \text{op} \rangle ('(\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{num} \rangle^*)'$
⇒ $\langle \text{expr} \rangle \langle \text{op} \rangle ('(\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle^*)'$
⇒ $\langle \text{expr} \rangle \langle \text{op} \rangle ('(\langle \text{expr} \rangle^*)'$
⇒ $\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$
⇒ $\langle \text{expr} \rangle$

(Quedando de esta forma)

Ambigüedad



$$12 + (5 * 8) = 52$$

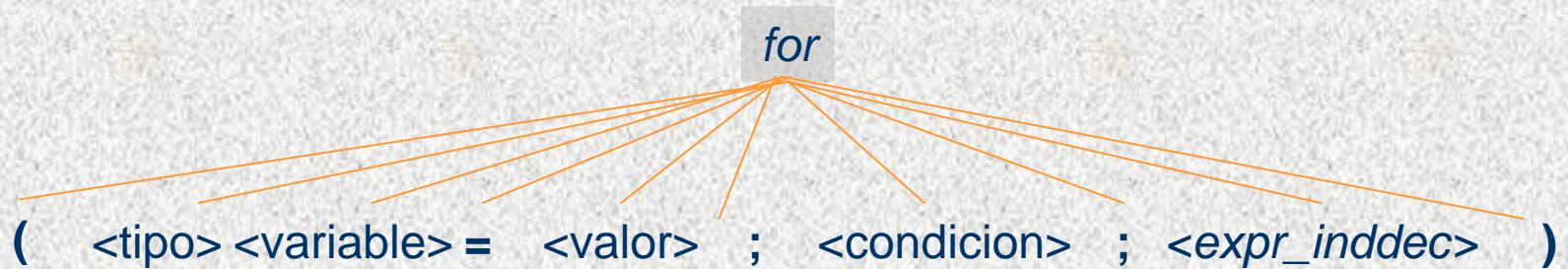


$$(12 + 5) * 8 = 136$$

Dos o más derivaciones distintas para una misma expresión

Ejemplo de Arbol de Derivación

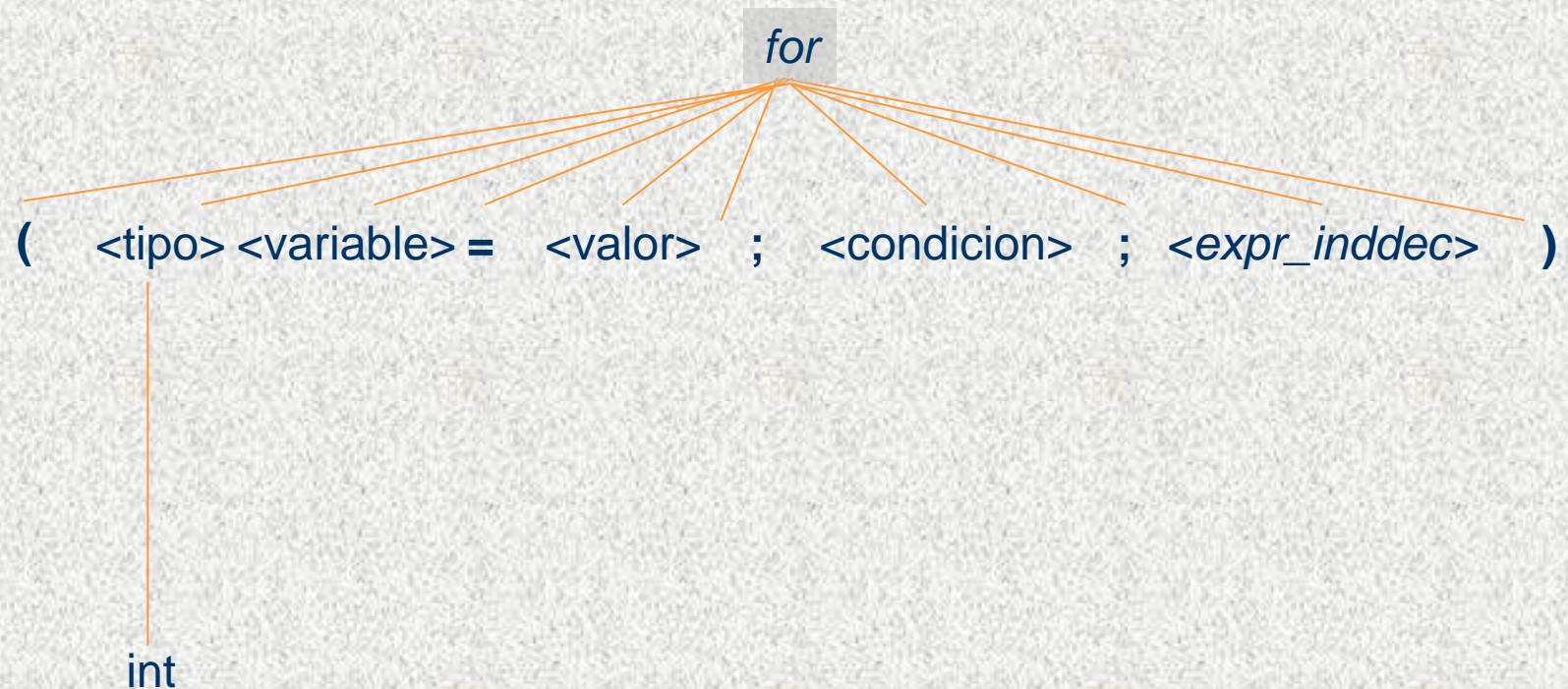
`<for> ::= '[' {<tipo> <variable> '=' <valor>}]; '['{ <condicion>}]; '[',{<expr_incdec>}''`



for(int ind = 0; ind < 10; ind++)

Ejemplo

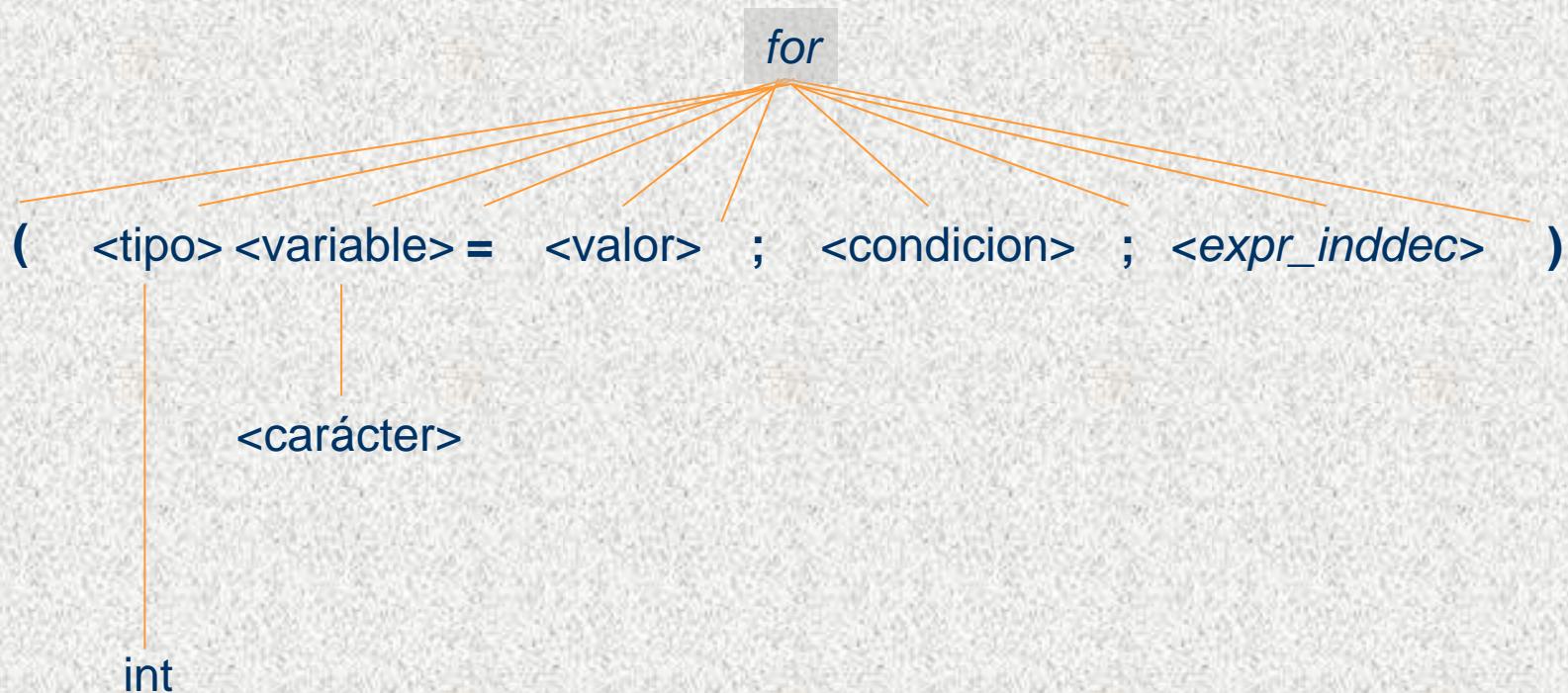
```
<for> ::= '[' {<tipo> <variable> '=' <valor>} ]';' [ ,'{ <condicion>} ]';' [, '{<expr_incdec>} ]'  
<tipo> ::= 'int' | 'char' | 'float' | 'double' | 'long' | 'short' | 'void'
```



for(int ind = 0; ind < 10; ind++)

Ejemplo

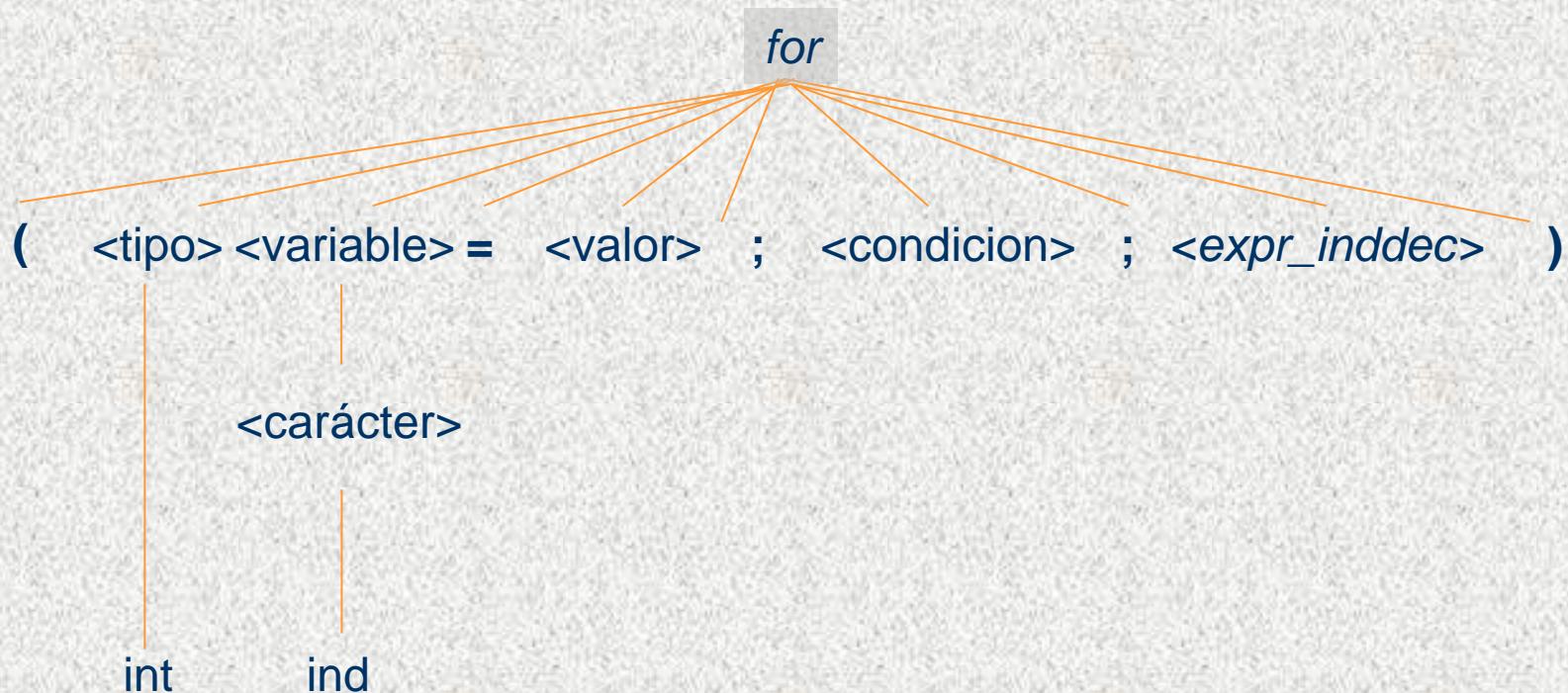
```
<for> ::= '[' {<tipo> <variable> '=' <valor>} ]';' [ ,'{ <condicion>} ]';' [,',{<expr_incdec>} ]'  
<tipo> ::= 'int' | 'char' | 'float' | 'double' | 'long' | 'short' | 'void'  
<variable> ::= <carácter>+  
<caracter> ::= 'a' | 'b' |.... | 'z' | 'A' |.... | 'Z'
```



for(int ind = 0; ind < 10; ind++)

Ejemplo

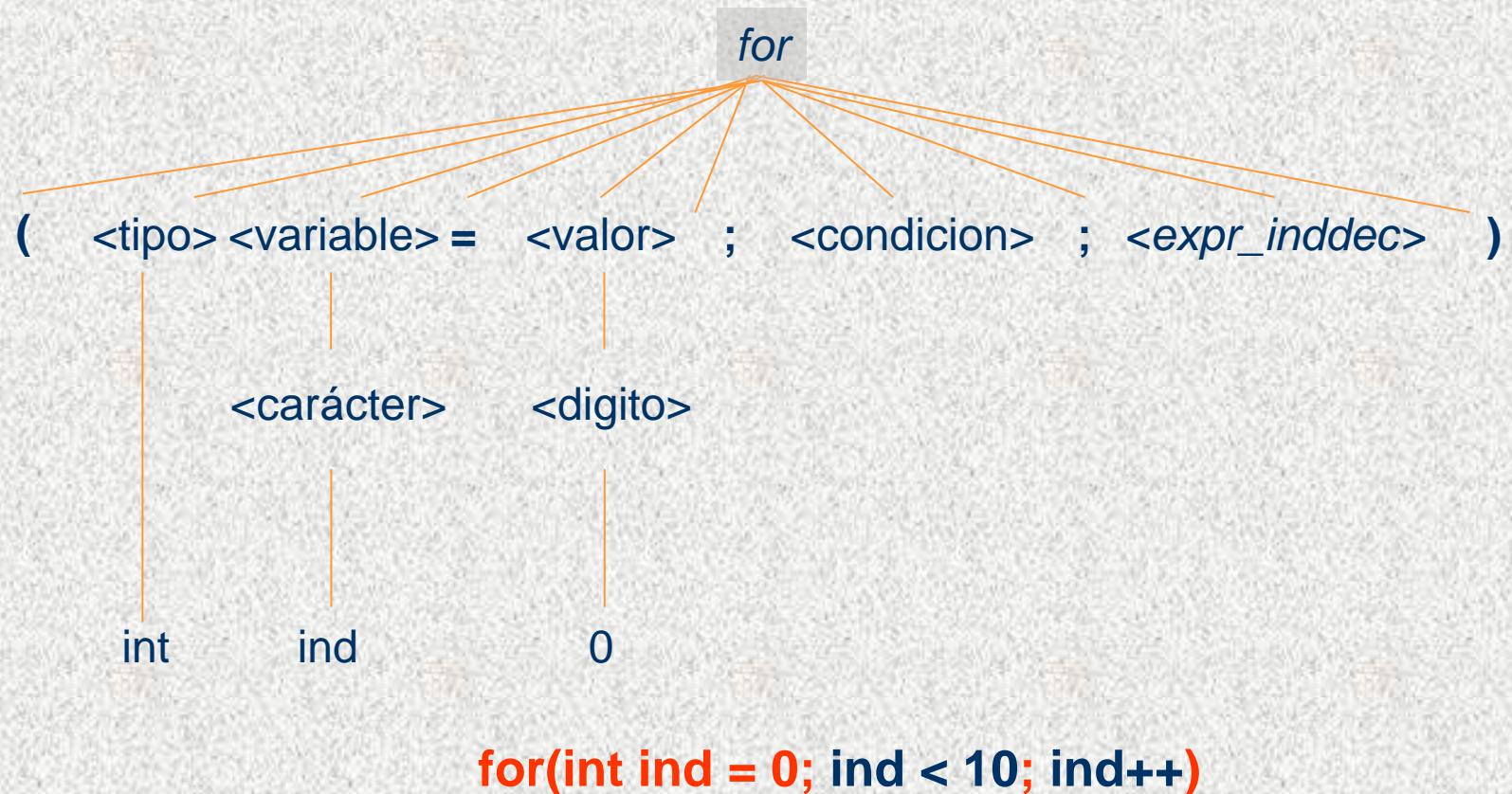
```
<for> ::= '[' {<tipo> <variable> '=' <valor>} ]';' [', '{ <condicion>} ]';' [, '{<expr_incdec>} ]'  
<tipo> ::= 'int' | 'char' | 'float' | 'double' | 'long' | 'short' | 'void'  
<variable> ::= <carácter>+  
<carácter> ::= 'a' | 'b' |.... | 'z' | 'A' |.... | 'Z'
```



for(int ind = 0; ind < 10; ind++)

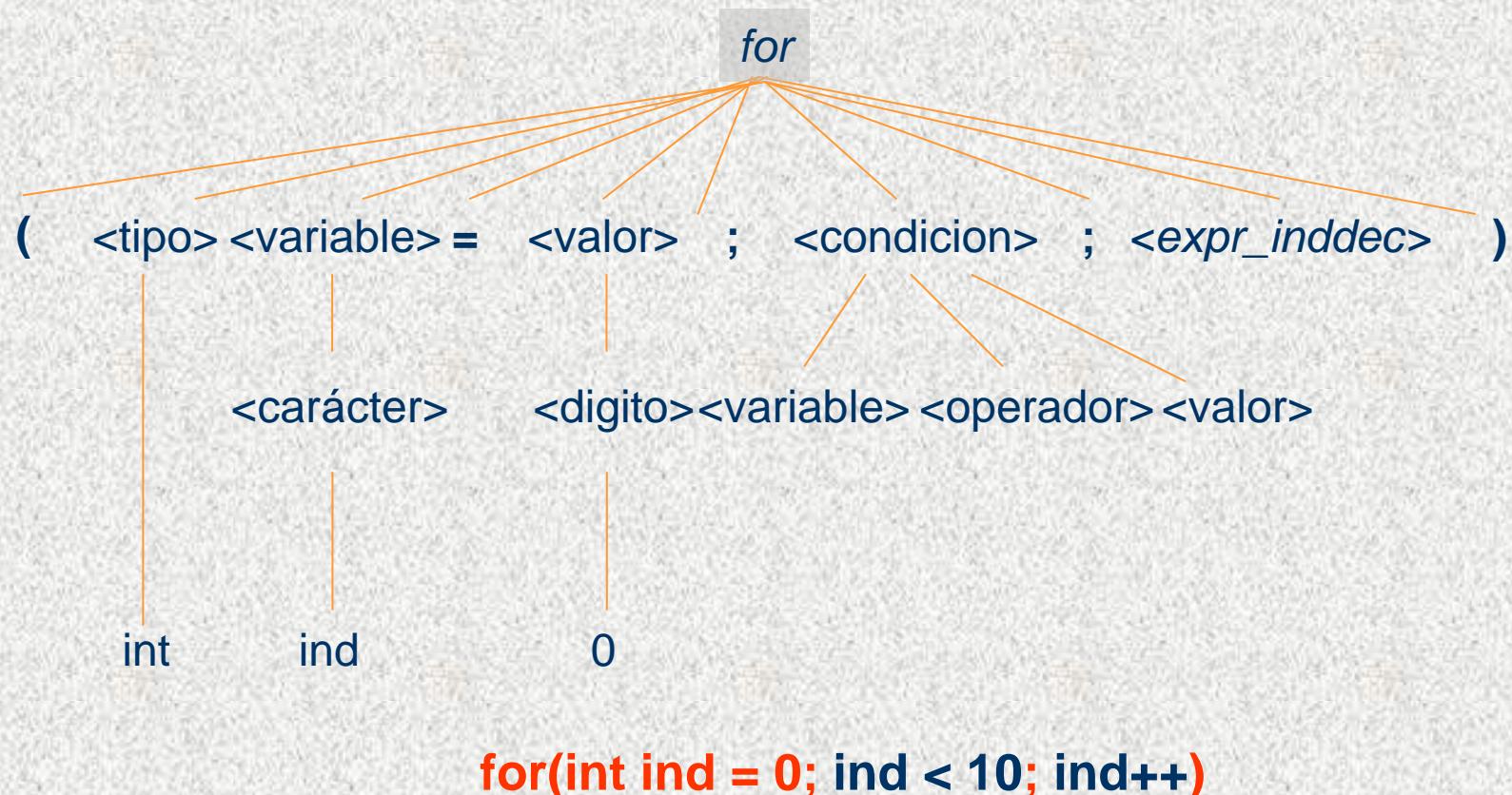
Ejemplo

```
<for> ::= '[' {<tipo> <variable> '=' <valor>} ]';' [ ,'{ <condicion>} ]';' [,',{<expr_incdec>} ]'  
<tipo> ::= 'int' | 'char' | 'float' | 'double' | 'long' | 'short' | 'void'  
<variable> ::= {<carácter>}+  
<carácter> ::= 'a' | 'b' |.... | 'z' | 'A' |.... | 'Z'  
<valor> ::= {<dígito>}+  
<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```



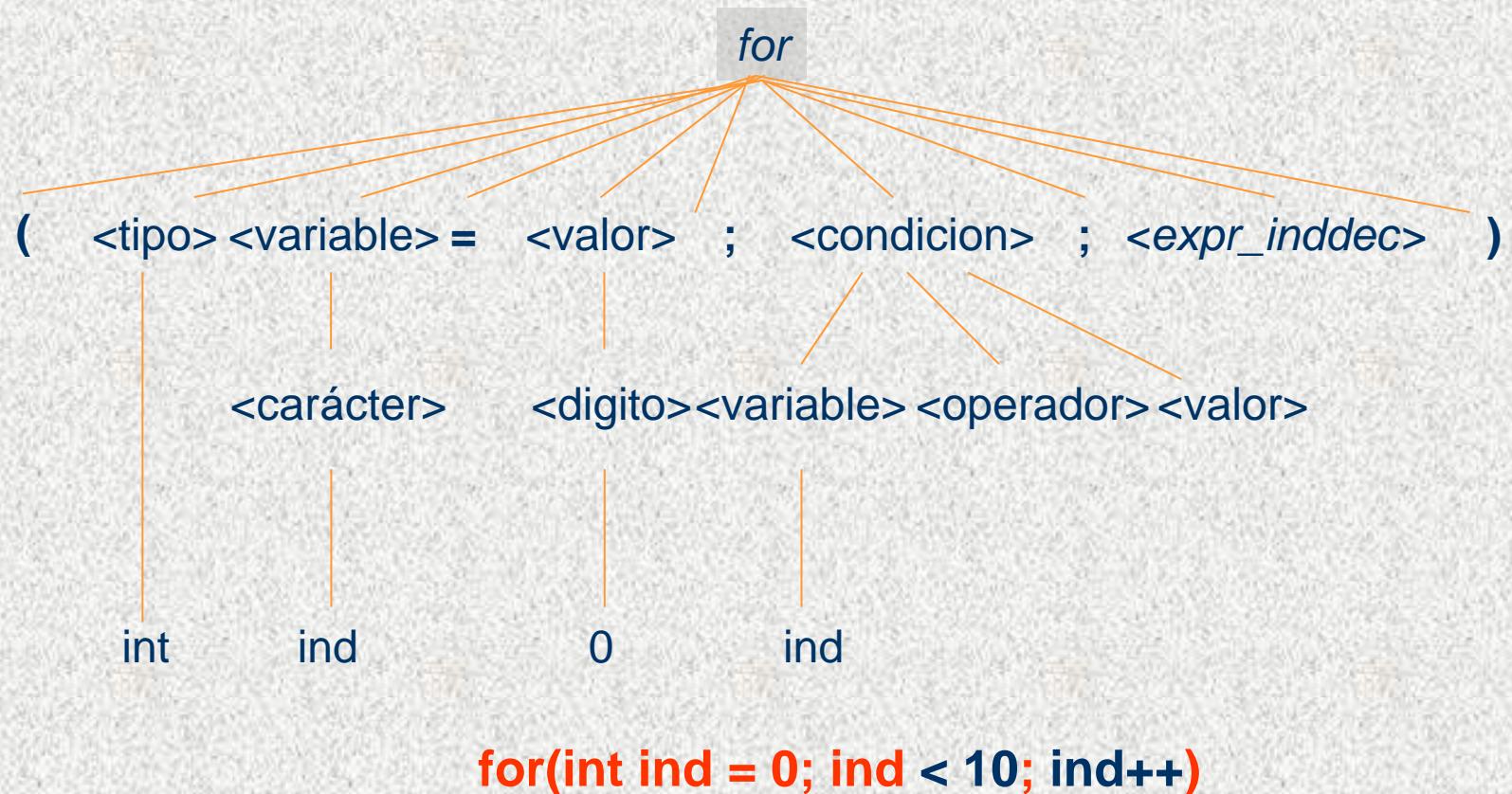
Ejemplo

```
<for> ::= '[' {<tipo> <variable> '=' <valor>} ]';' [ ,'{ <condicion>} ]';' [,'{<expr_incdec>} ]'  
<tipo> ::= 'int' | 'char' | 'float' | 'double' | 'long' | 'short' | 'void'  
<variable> ::= {<carácter>}+  
<caracter> ::= 'a' | 'b' |.... | 'z' | 'A' |.... | 'Z'  
<valor> ::= {<dígito>}+  
<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  
<condicion> ::= <variable><operador><valor>
```



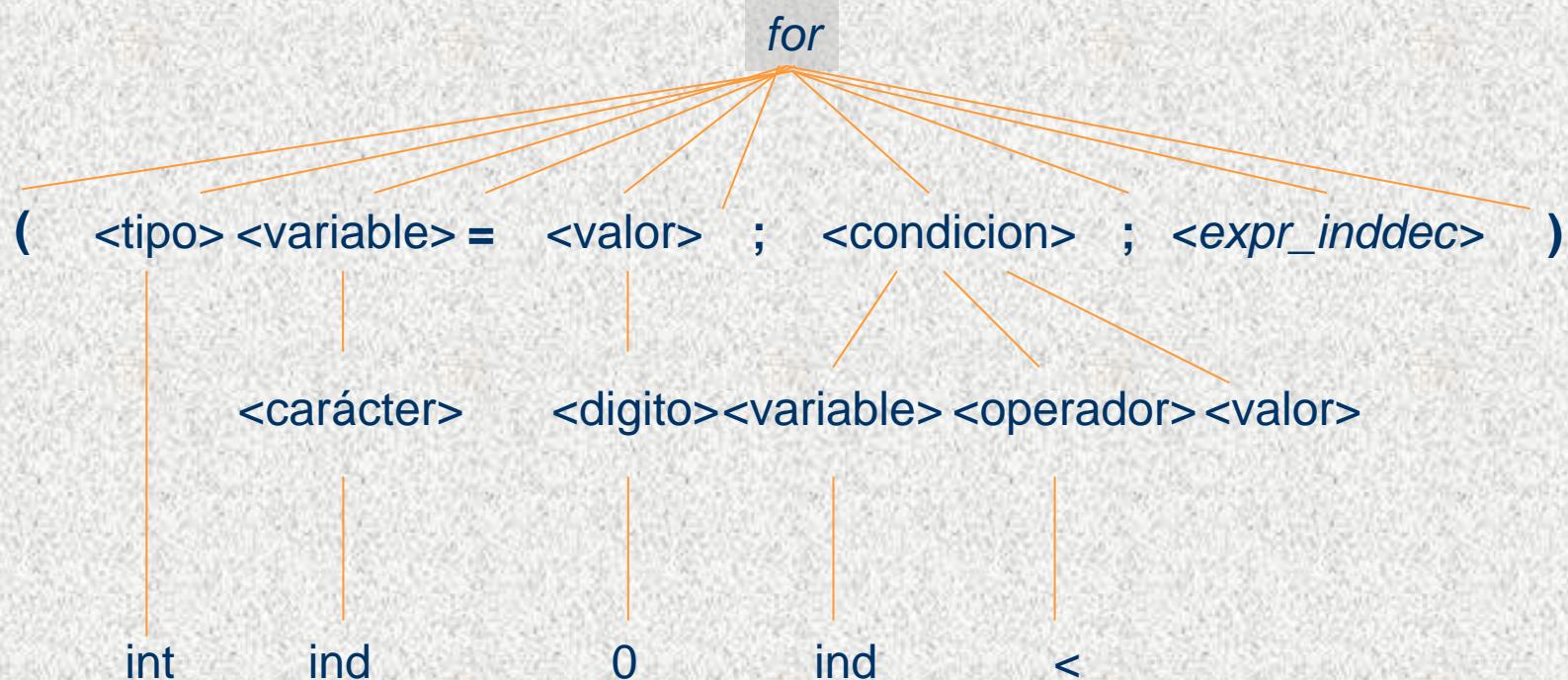
Ejemplo

```
<for> ::= '[' {<tipo> <variable> '=' <valor>} ]';' [ ,'{ <condicion>} ]';' [,'{<expr_incdec>} ]'  
<tipo> ::= 'int' | 'char' | 'float' | 'double' | 'long' | 'short' | 'void'  
<variable> ::= {<carácter>}+  
<caracter> ::= 'a' | 'b' |.... | 'z' | 'A' |.... | 'Z'  
<valor> ::= {<dígito>}+  
<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  
<condicion> ::= <variable><operador><valor>
```



Ejemplo

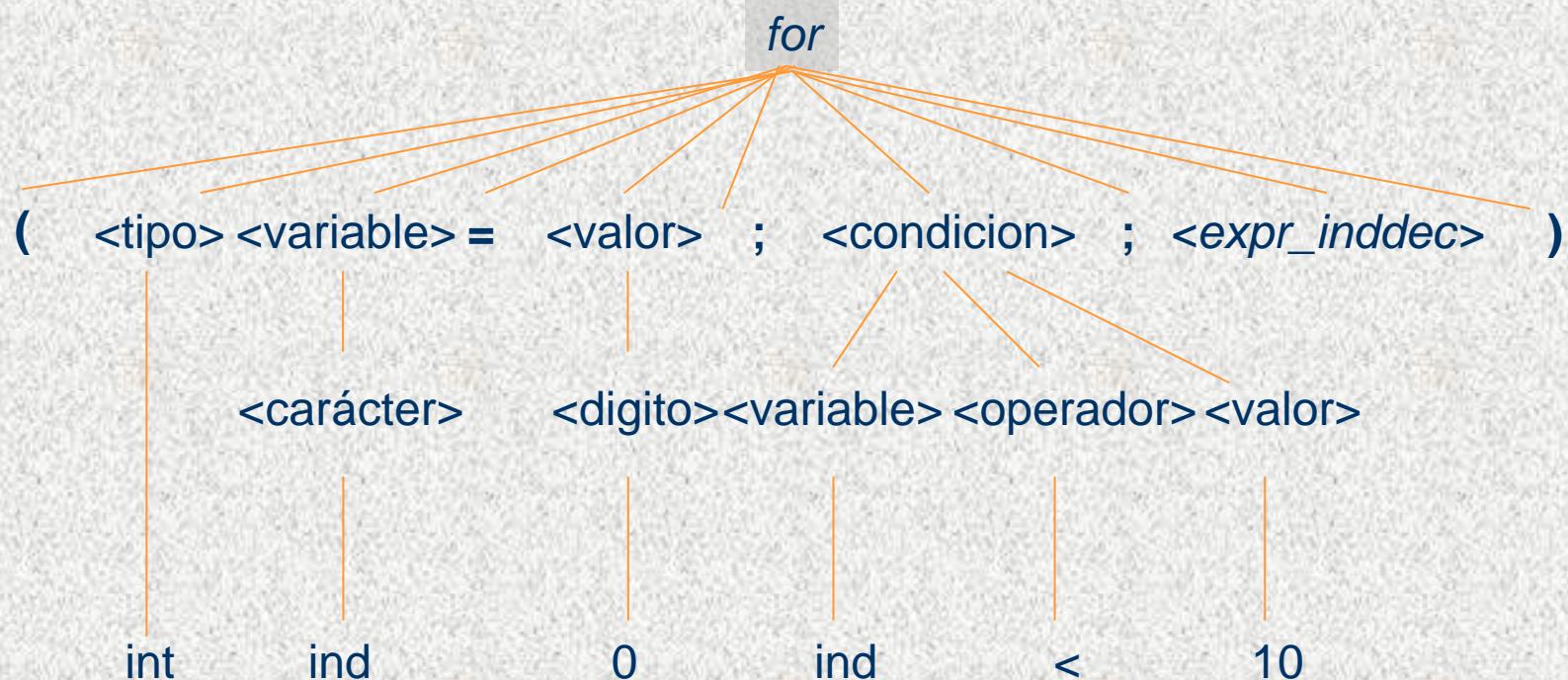
```
<for> ::= '[' {<tipo> <variable> '=' <valor>} ]';' [ ,'{ <condicion>} ]';' [,',{<expr_incdec>} ]'
<tipo> ::= 'int' | 'char' | 'float' | 'double' | 'long' | 'short' | 'void'
<variable> ::= {<carácter>}+
<carácter> ::= 'a' | 'b' |.... | 'z' | 'A' |.... | 'Z'
<valor> ::= {<dígito>}+
<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<condicion> ::= <variable><operador><valor>
<operador> ::= '+' | '-' | 'I' | '*' | '>' | '>=' | '<' | '<=' | '=' | '==' | '!'
```



for(int ind = 0; ind < 10; ind++)

Ejemplo

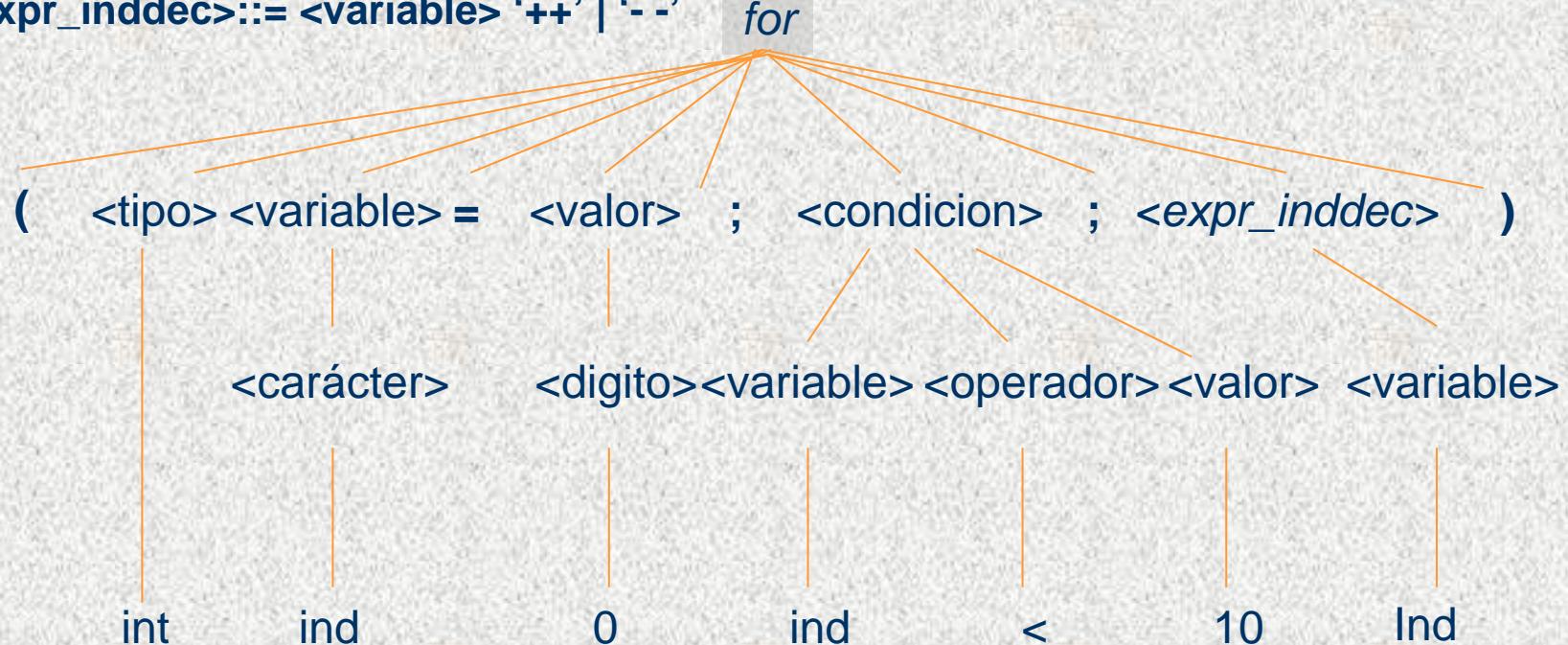
```
<for> ::= '[' {<tipo> <variable> '=' <valor>} ]';' [ ,'{ <condicion>} ]';' [,',{<expr_incdec>} ]'
<tipo> ::= 'int' | 'char' | 'float' | 'double' | 'long' | 'short' | 'void'
<variable> ::= {<carácter>}+
<carácter> ::= 'a' | 'b' |.... | 'z' | 'A' |.... | 'Z'
<valor> ::= {<dígito>}+
<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<condicion> ::= <variable><operador><valor>
<operador> ::= '+' | '-' | 'I' | '*' | '>' | '>=' | '<' | '<=' | '=' | '==' | '!='
```



for(int ind = 0; ind < 10; ind++)

Ejemplo

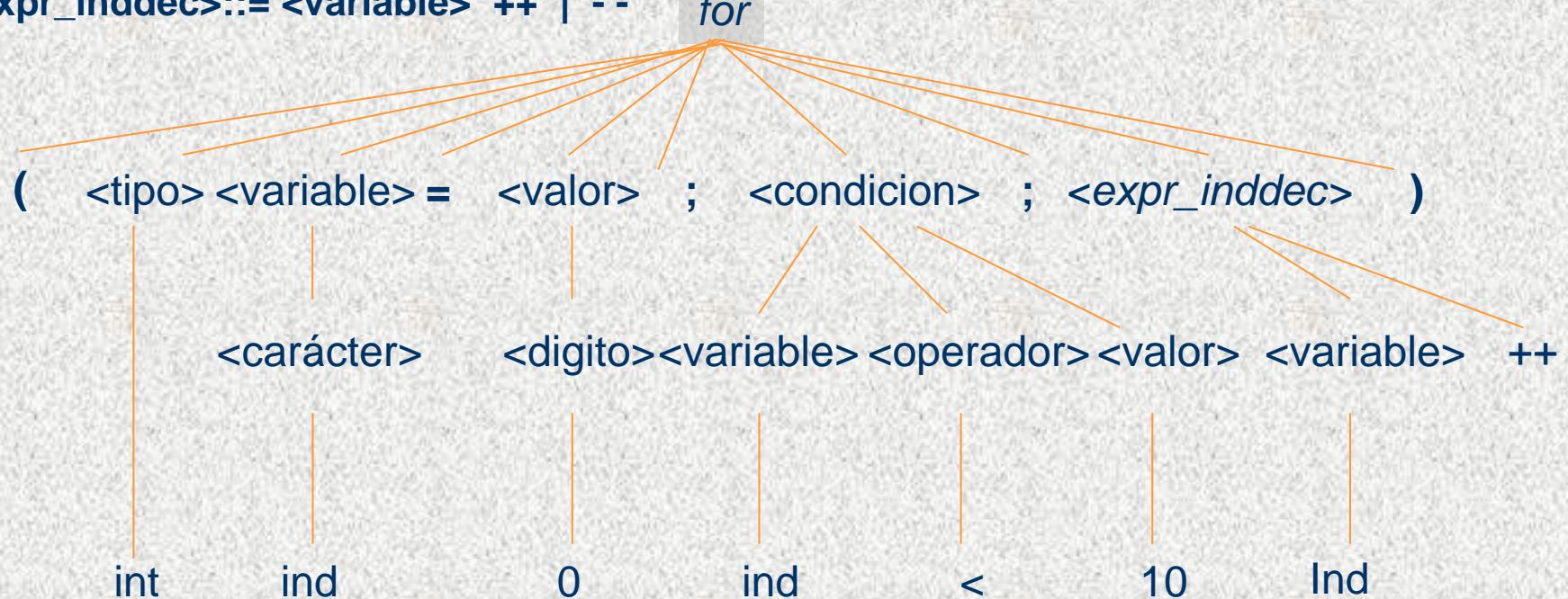
```
<for> ::= '[' {<tipo> <variable> '=' <valor>} ]';' [ ,'{ <condicion>} ]';' [,',{<expr_incdec>} ]'
<tipo> ::= 'int' | 'char' | 'float' | 'double' | 'long' | 'short' | 'void'
<variable> ::= {<carácter>}+
<caracter> ::= 'a' | 'b' |.... | 'z' | 'A' |.... | 'Z'
<valor> ::= {<dígito>}+
<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<condicion> ::= <variable><operador><valor>
<operador> ::= '+' | '-' | 'l' | '*' | '>' | '>=' | '<' | '<=' | '=' | '==' | '!'
<expr_incdec> ::= <variable> '++' | '--'
```



`for(int ind = 0; ind < 10; ind++)`

Ejemplo

```
<for> ::= '[' {<tipo> <variable> '=' <valor>} ]';' [ ,'{ <condicion>} ]';' [,',{<expr_incdec>} ]'
<tipo> ::= 'int' | 'char' | 'float' | 'double' | 'long' | 'short' | 'void'
<variable> ::= {<carácter>}+
<caracter> ::= 'a' | 'b' |.... | 'z' | 'A' |.... | 'Z'
<valor> ::= {<dígito>}+
<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<condicion> ::= <variable><operador><valor>
<operador> ::= '+' | '-' | 'l' | '*' | '>' | '>=' | '<' | '<=' | '=' | '==' | '!'
<expr_incdec> ::= <variable> '++' | '--'
```



for(int ind = 0; ind < 10; ind++) !!! Cadena aceptada !!!