# 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 Contexto

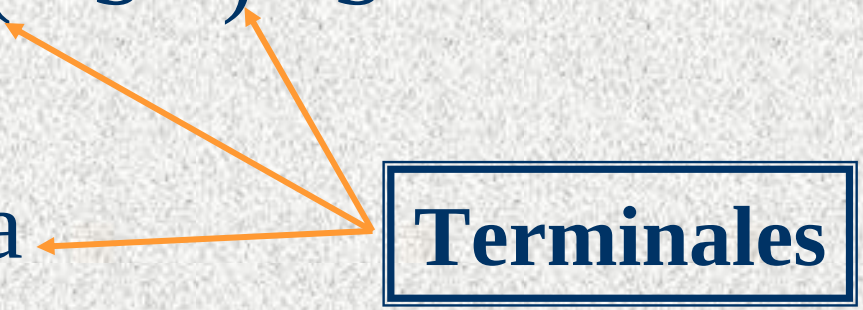
<S> ::= ( <S> ) <S>

<S> ::= a

# Gramática Libre de Contexto

<S> ::= ( <S> ) <S>

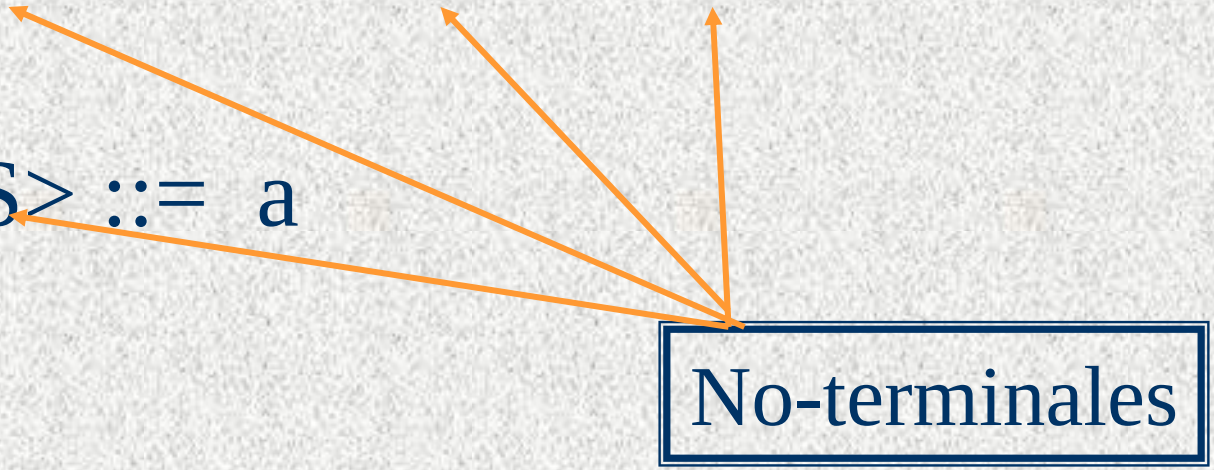
<S> ::= a



# Gramática Libre de Contexto

<S> ::= ( <S> ) <S>

<S> ::= a



# Gramática Libre de Contexto

<S> ::= ( <S> ) <S>

<S> ::= a

Símbolo inicial: <S>

# Gramática Libre de Contexto

<S> ::= ( <S> ) <S>

<S> ::= a

Regla o Producción

# Gramática Libre de Contexto

<expr> ::= <expr> <op> <expr>

<expr> ::= ( <expr> ) | <expr>

<expr> ::= - <expr>

<expr> ::= <num>

<op> ::= +

<op> ::= *

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

<num> ::= <dig> | <num>

<dig> ::= 0 | 1| 2| 3| 4| 5| 6| 7| 8| 9

# Gramática Libre de Contexto

<expr> ::= <expr> <op> <expr>

<expr> ::= ( <expr> ) | <expr>

<expr> ::= - <expr>

<expr> ::= <num>

<op> ::= +

<op> ::= *

<num> ::= <dig> | <num>

<dig> ::= 0 | 1| 2| 3| 4| 5| 6| 7| 8| 9

**No Terminales:**

**expr, op, num, dig**

# Gramática Libre de Contexto

<expr> ::= <expr> <op> <expr>

<expr> ::= ( <expr> ) | <expr>

<expr> ::= - <expr>

<expr> ::= <num>

<op> ::= +

<op> ::= *

<num> ::= <dig> | <num>

<dig> ::= 0 | 1| 2| 3| 4| 5| 6| 7| 8| 9

**Símbolo inicial:**

**expr**

# Gramática Libre de Contexto

<expr> ::= <expr> <op> <expr>

<expr> ::= ( <expr> ) | <expr>

<expr> ::= - <expr>

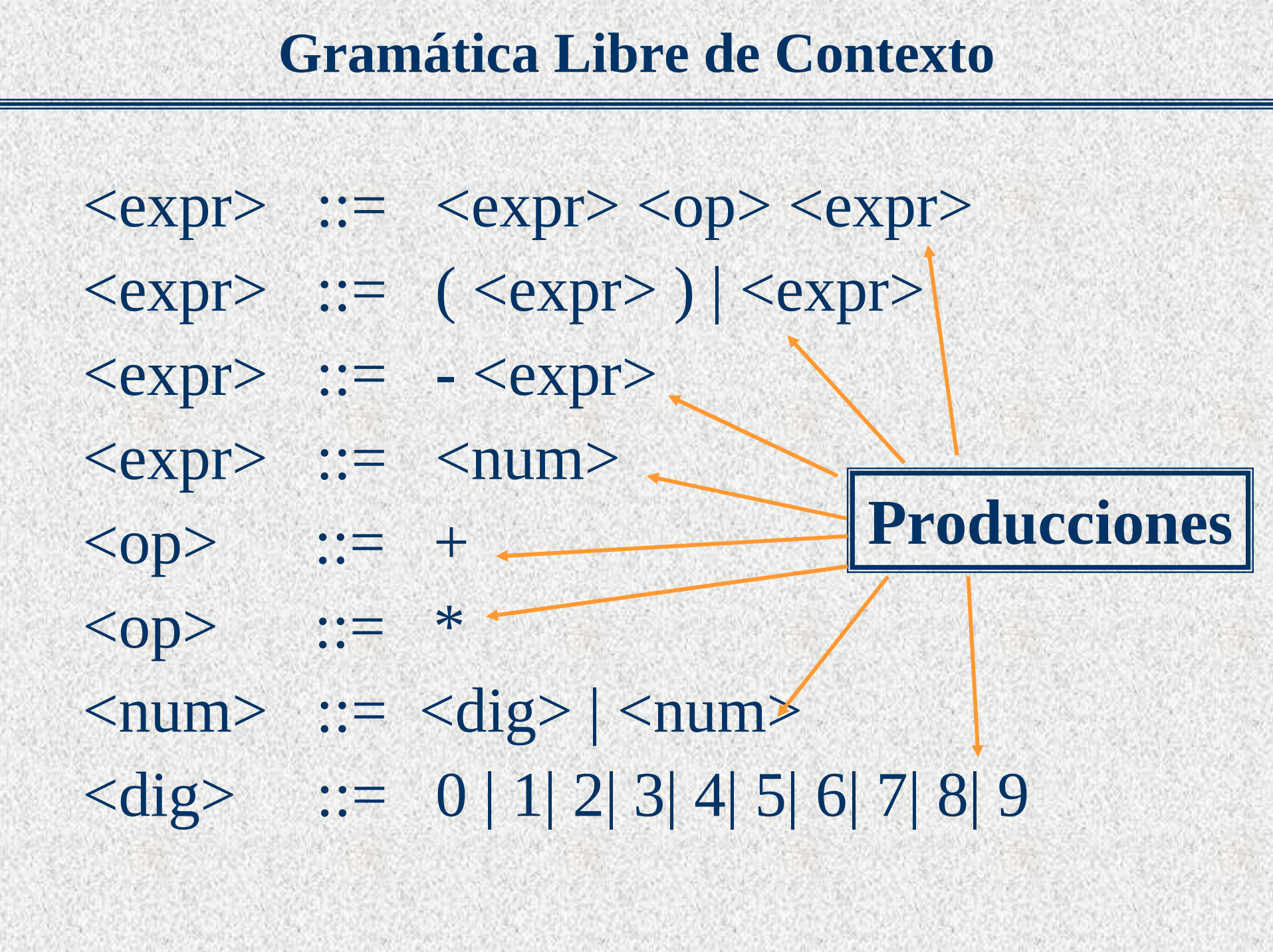
<expr> ::= <num>

<op> ::= +

<op> ::= *

<num> ::= <dig> | <num>

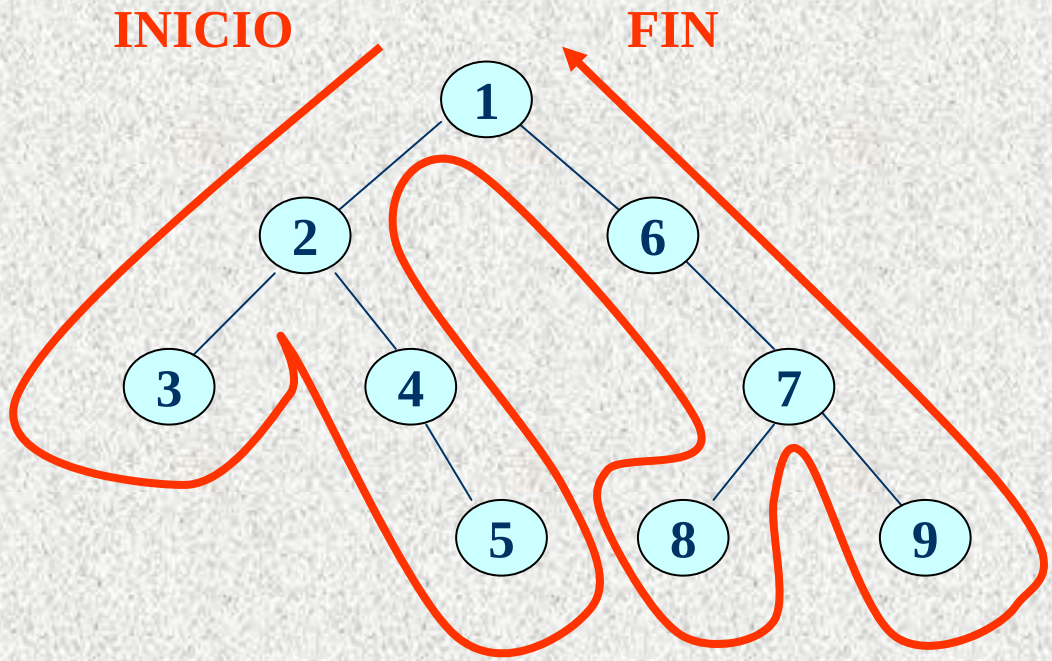
<dig> ::= 0 | 1| 2| 3| 4| 5| 6| 7| 8| 9



# Recordatorio

## (Recorridos)

## Recorrido Pre-Orden (Derivación izquierda)

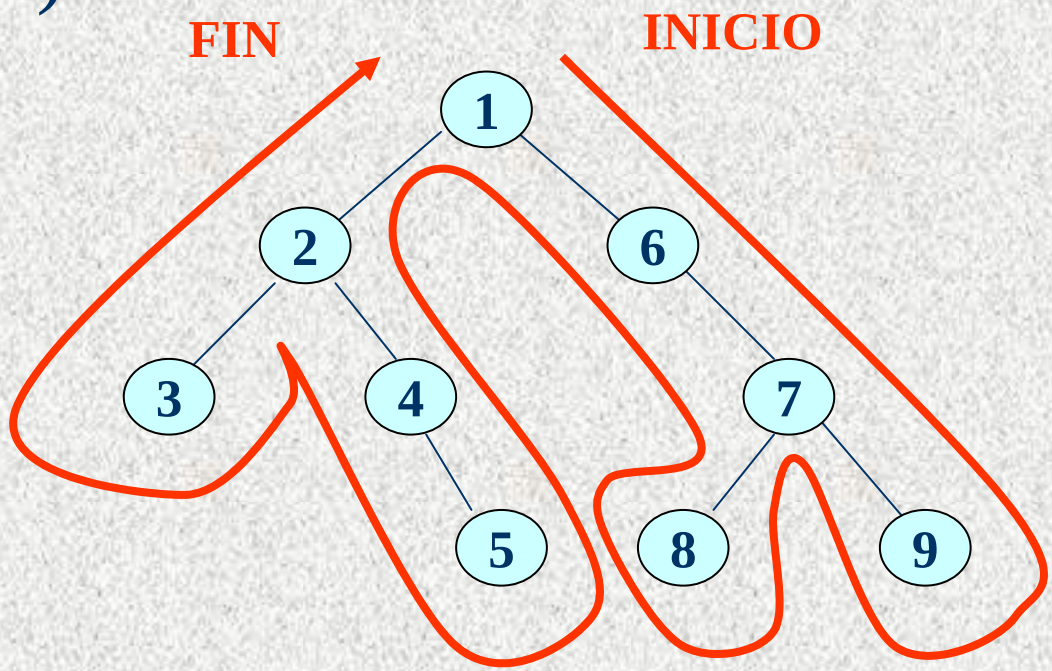



**Recorrido: 1,2,3,4,5,6,7,8,9**

# Recordatorio

## (Recorridos)

Recorrido POS-Orden-Inverso (Derivación derecha)

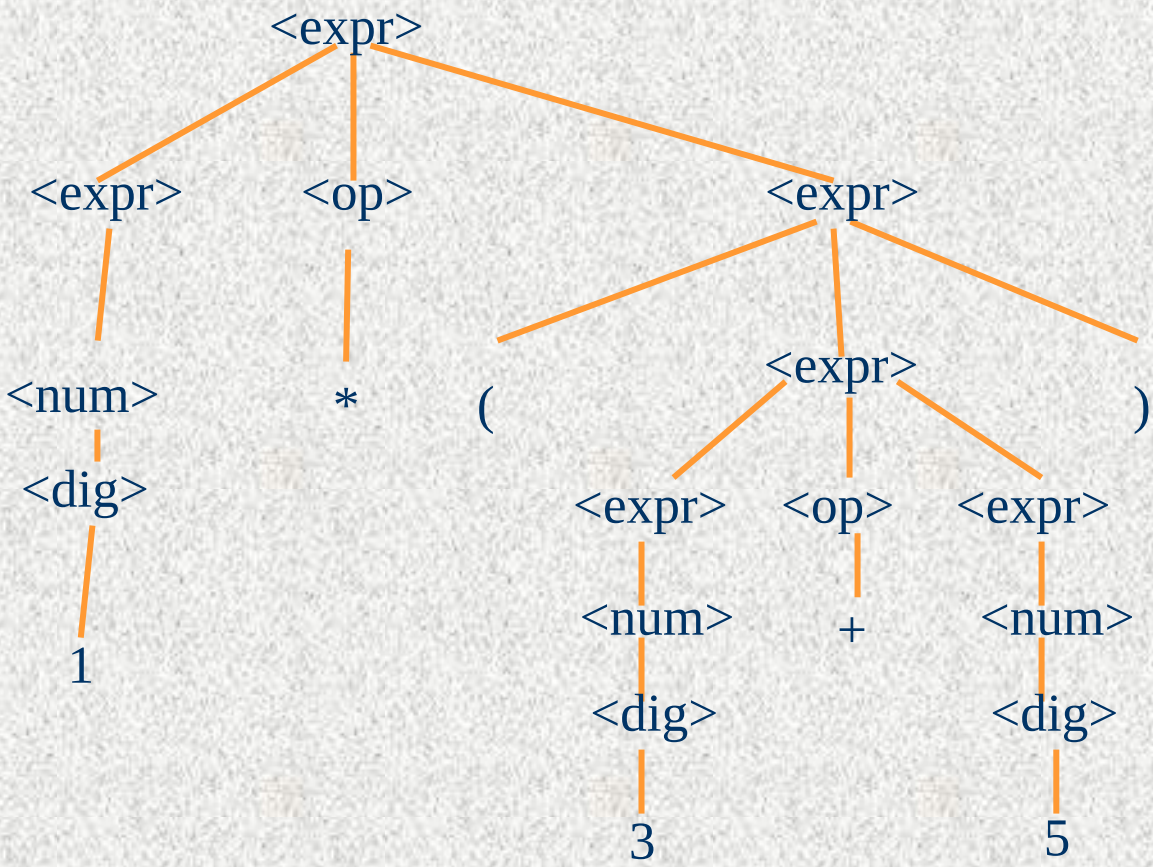



**Recorrido: 1,6,7,9,8,2,4,5,3**

# Arbol de Parseo
# Derivación por la izquierda

<expr> ::= <num>

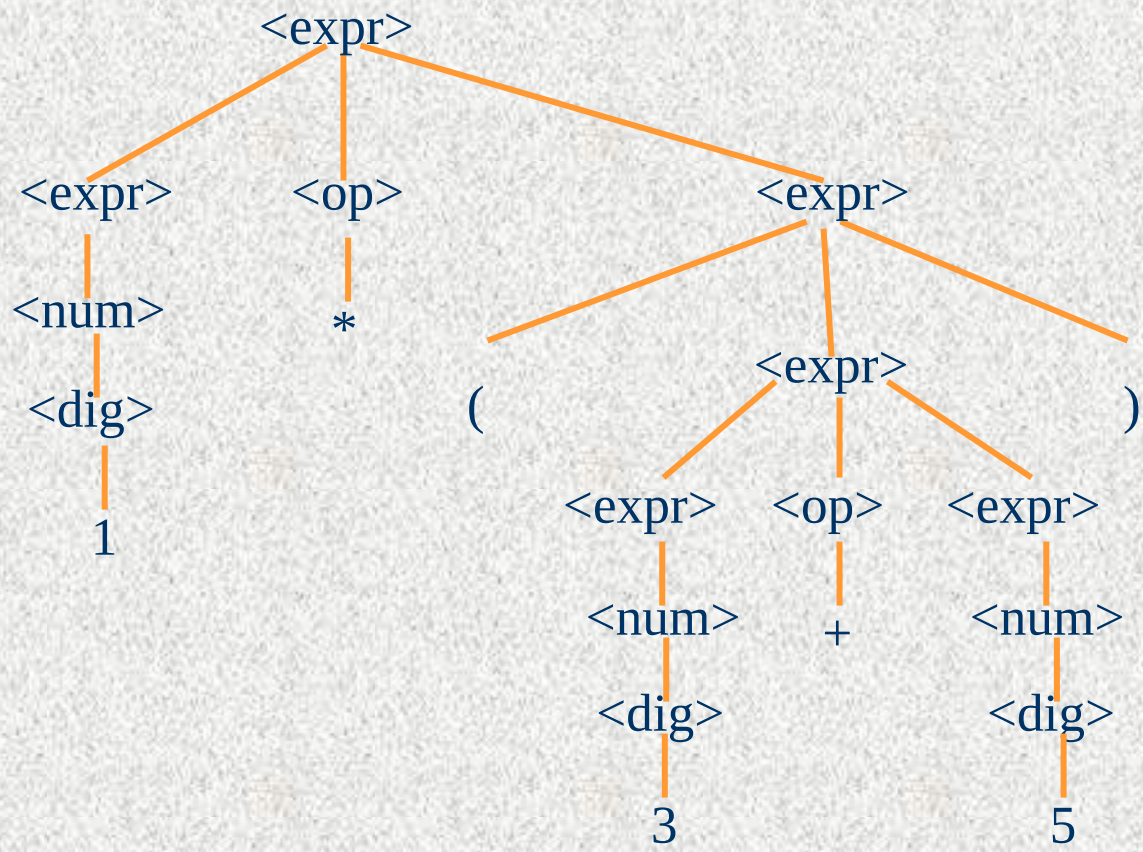
Cadena: 1 ‘*’ ‘(‘ 3 ‘+’ 5 ‘)’



# Arbol de Parseo
# Derivación por la derecha

Producción: <expr> ⇒ 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)

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

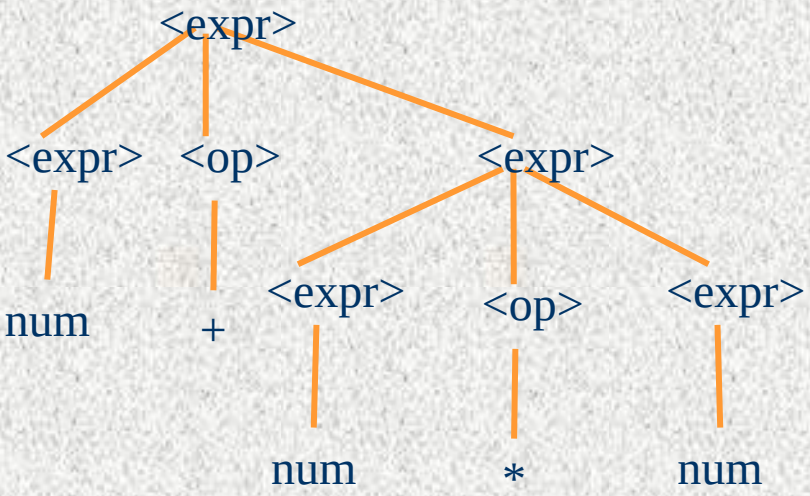
**(Se invierte el orden de evaluación)**

# Parser ‘Bottom-Up’ (LR)

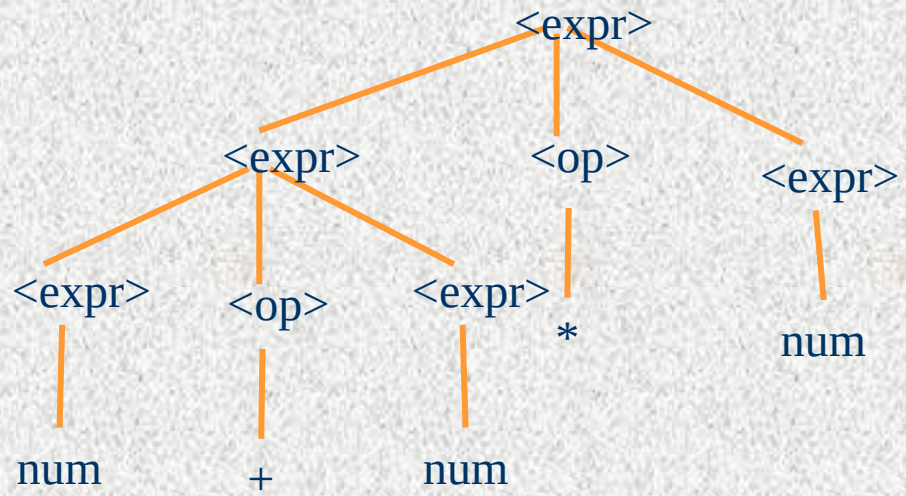
⇒ <dig> ‘*’ ‘(‘ <dig> ‘+’ <dig> ‘)’
⇒ <num> ‘*’ ‘(‘ <dig> ‘+’ <dig> ‘)’
⇒ <expr> ‘*’ ‘(‘ <dig> ‘+’ <dig> ‘)’
⇒ <expr> <op> ‘(‘ <dig> ‘+’ <dig> ‘)’
⇒ <expr> <op> ‘(‘ <num> ‘+’ <dig> ‘)’
⇒ <expr> <op> ‘(‘ <expr> ‘+’ <dig> ‘)’’
⇒ <expr> <op> ‘(‘ <expr> <op> <dig> ‘)’
⇒ <expr> <op> ‘(‘ <expr> <op> <num>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

**(Quedando de esta forma)**

# Ambigüedad




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

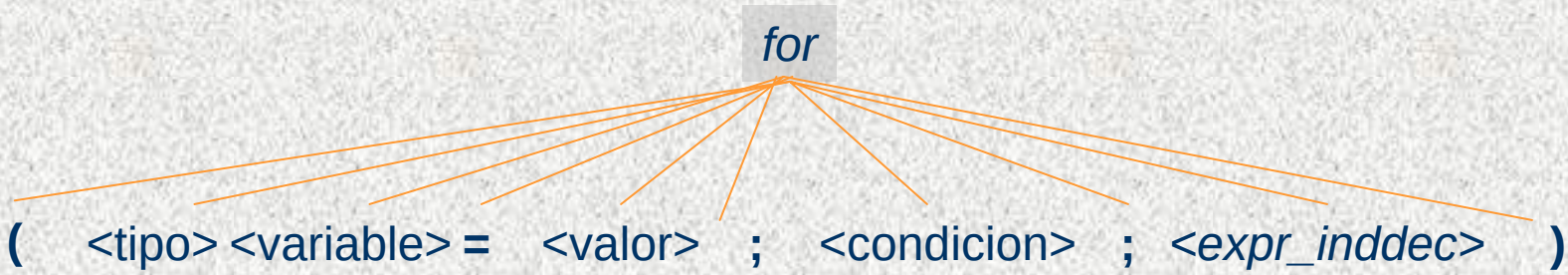



**(12 + 5) * 8 = 136**

Dos o más derivaciones distintas para una misma expresión

# Ejemplo de Arbol de Derivación

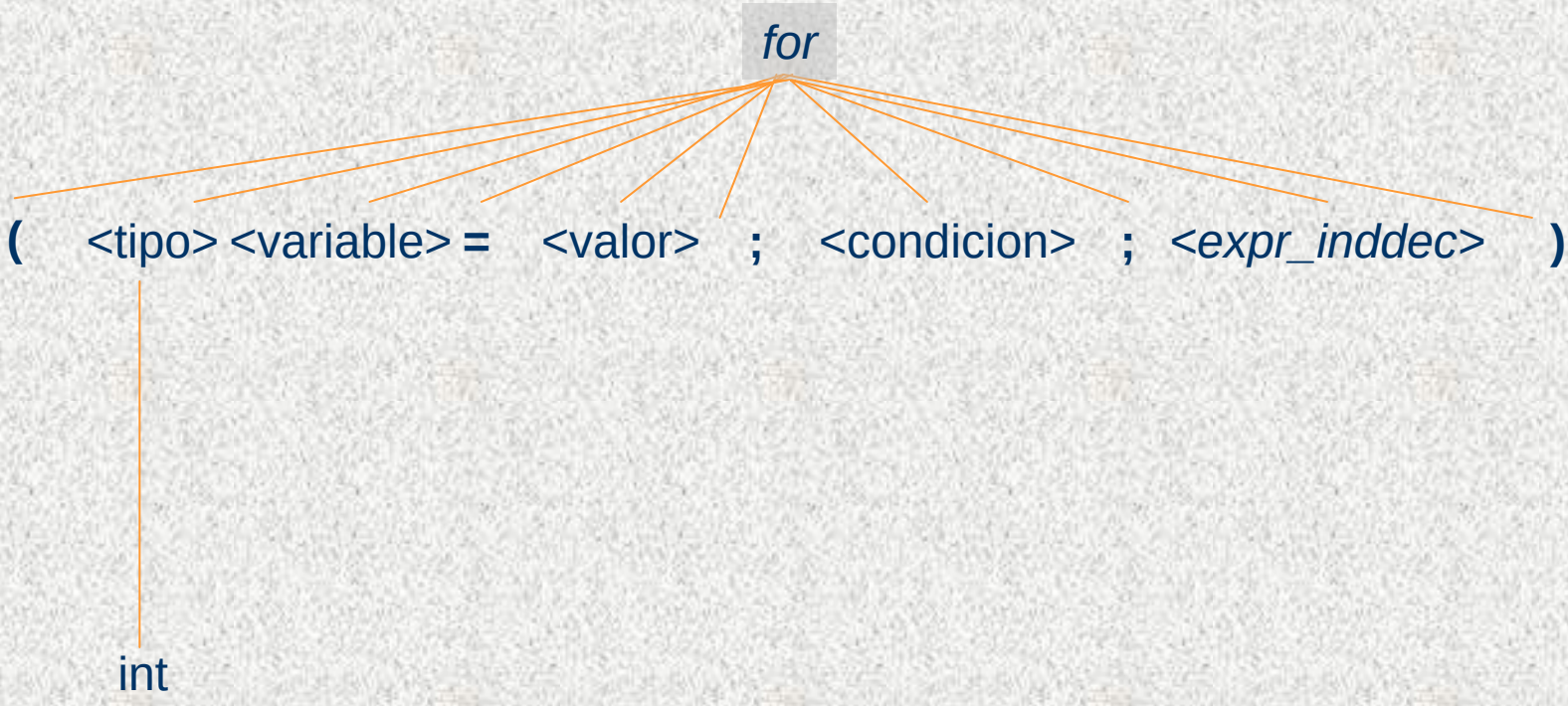
**<for>::= ‘(‘[‘,’ {<tipo> <variable> ‘=‘ <valor>}]’;’[‘,’{ <condicion>}]’;’ [‘,’{<expr_incdec>}‘)’**




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

# Ejemplo

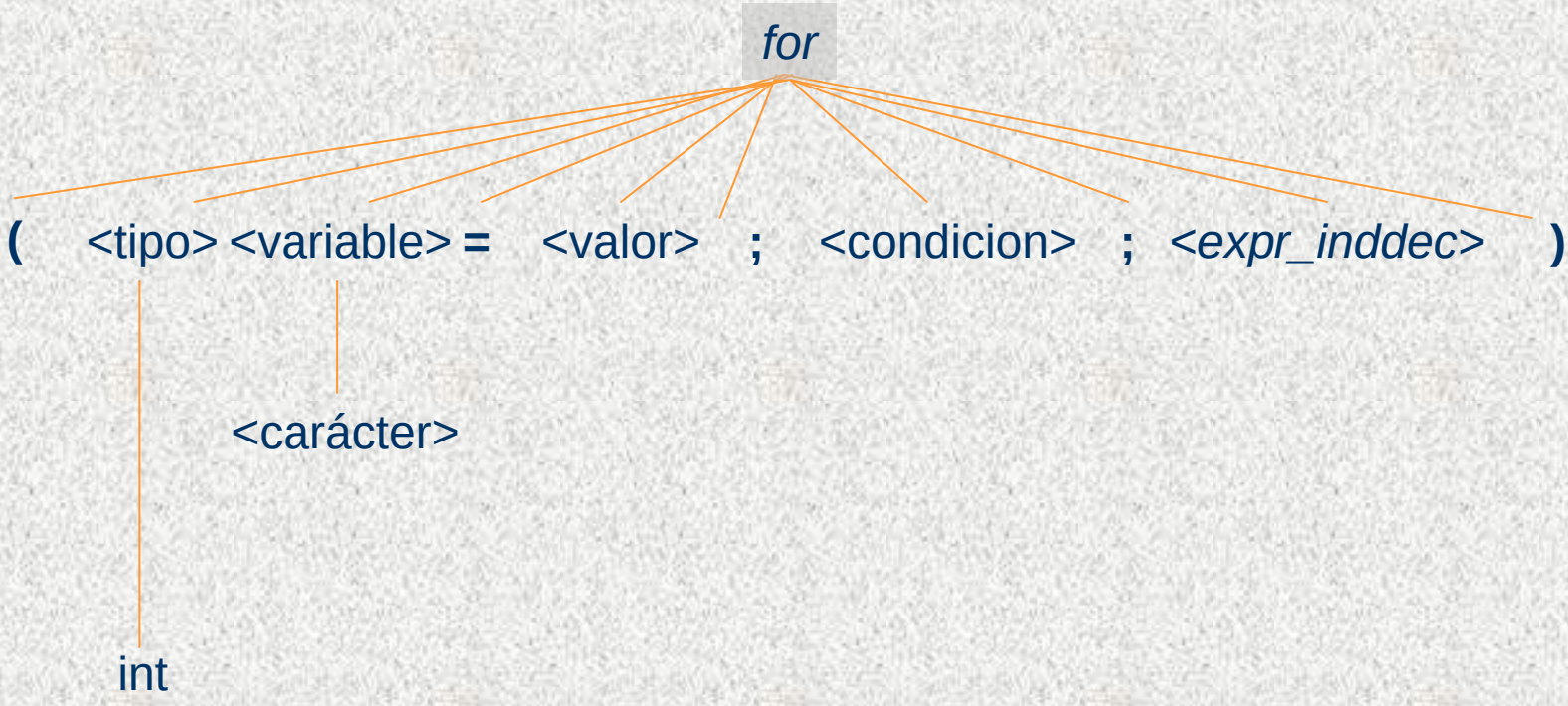
**<for>::= ‘(‘[‘,’ {<tipo> <variable> ‘=‘ <valor>}]’;’[‘,’{ <condicion>}]’;’ [‘,’{<expr_incdec>}‘)’**
**<tipo>::= ‘int’ | ‘char’ | ‘float’ | ‘double’ | ‘long’ | ‘short’ | ‘void’**




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

# Ejemplo

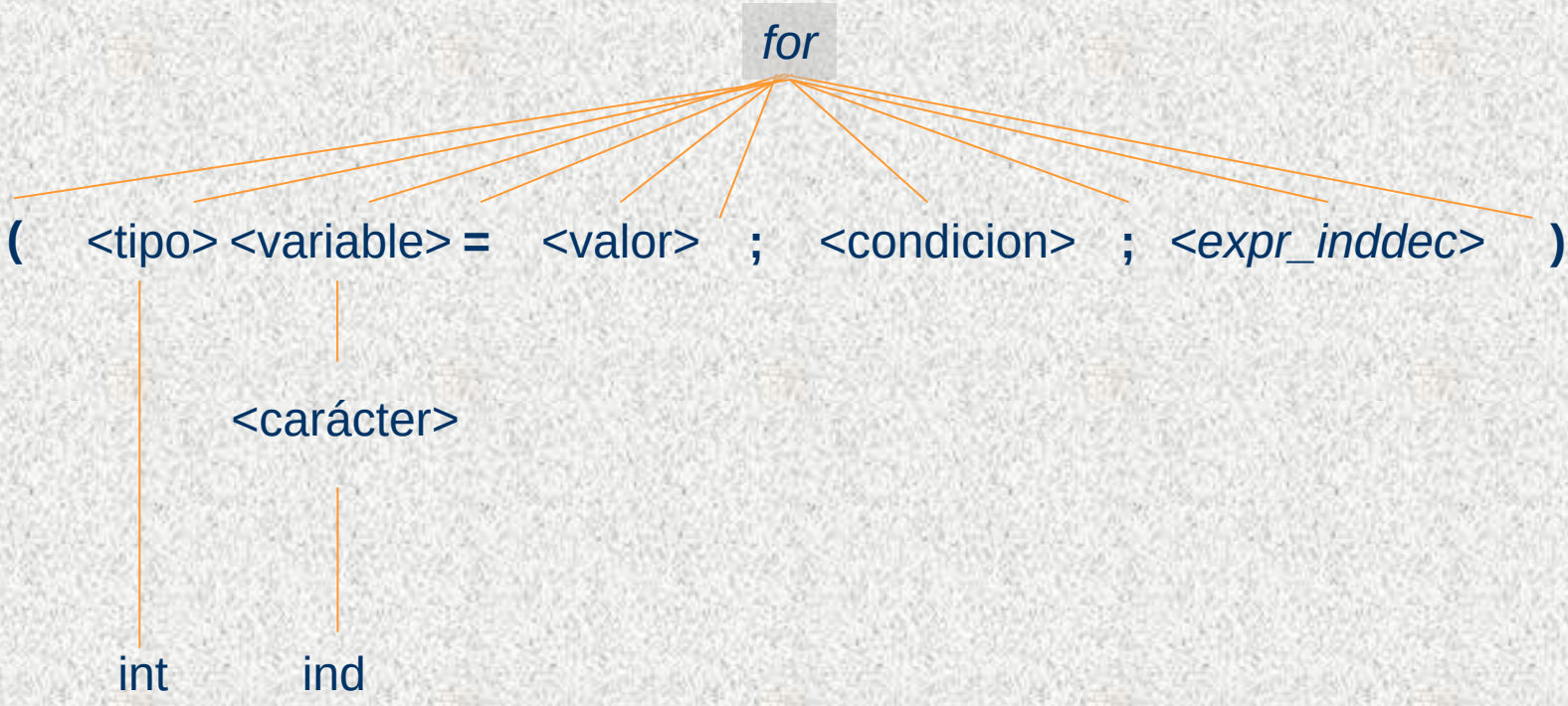
**<for>::= ‘(‘[‘,’ {<tipo> <variable> ‘=‘ <valor>}]’;’[‘,’{ <condicion>}]’;’ [‘,’{<expr_incdec>}‘)’**
**<tipo>::= ‘int’ | ‘char’ | ‘float’ | ‘double’ | ‘long’ | ‘short’ | ‘void’**
**<variable>::= {<carácter>}$^{+}$**
***<caracter>::= ‘a’* | ‘b’ |.... | ‘z’ | ‘A’ |.... | ‘Z’**




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

# Ejemplo

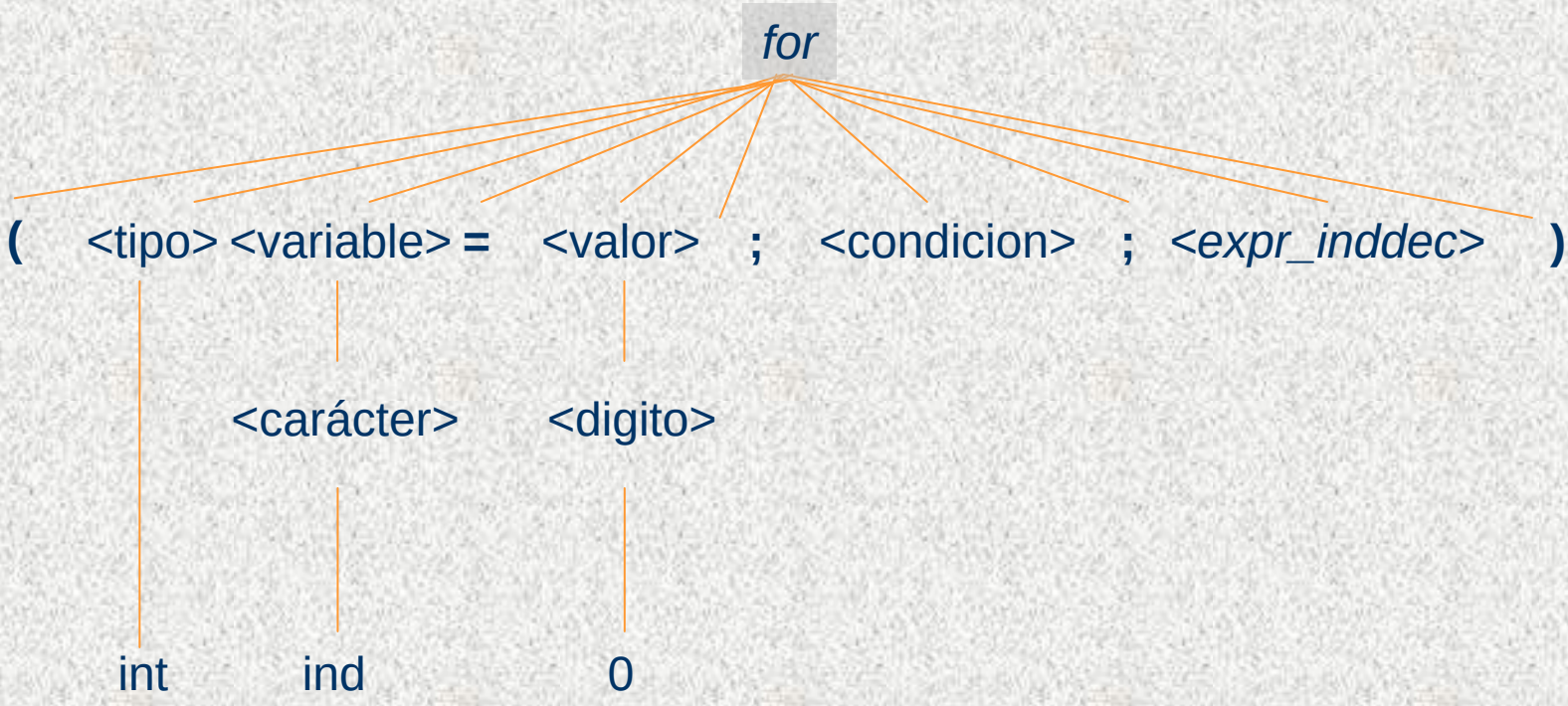
**<for>::= ‘(‘[‘,’ {<tipo> <variable> ‘=‘ <valor>}]’;’[‘,’{ <condicion>}]’;’ [‘,’{<expr_incdec>}‘)’**
**<tipo>::= ‘int’ | ‘char’ | ‘float’ | ‘double’ | ‘long’ | ‘short’ | ‘void’**
**<variable>::= {<carácter>}$^{+}$**
***<caracter>::= ‘a’* | ‘b’ |.... | ‘z’ | ‘A’ |.... | ‘Z’**




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

# Ejemplo

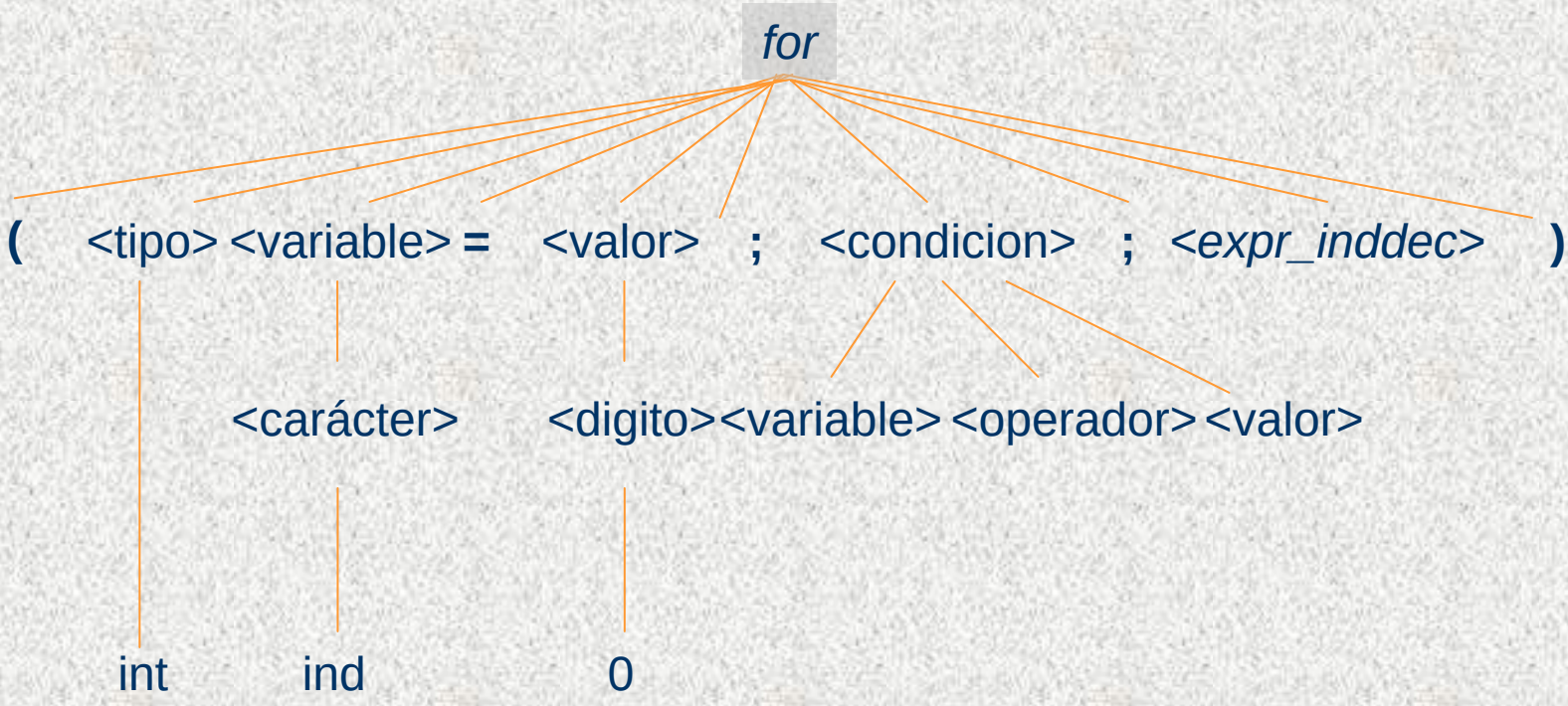
**<for>::= ‘(‘[‘,’ {<tipo> <variable> ‘=‘ <valor>}]’;’[‘,’{ <condicion>}]’;’ [‘,’{<expr_incdec>}‘)’**
**<tipo>::= ‘int’ | ‘char’ | ‘float’ | ‘double’ | ‘long’ | ‘short’ | ‘void’**
**<variable>::= {<carácter>}$^{+}$**
***<caracter>::= ‘a’* | ‘b’ |.... | ‘z’ | ‘A’ |.... | ‘Z’**
**<valor>::= {<digito>}** $^{+}$
***<digito>::= 0* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**




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

# Ejemplo

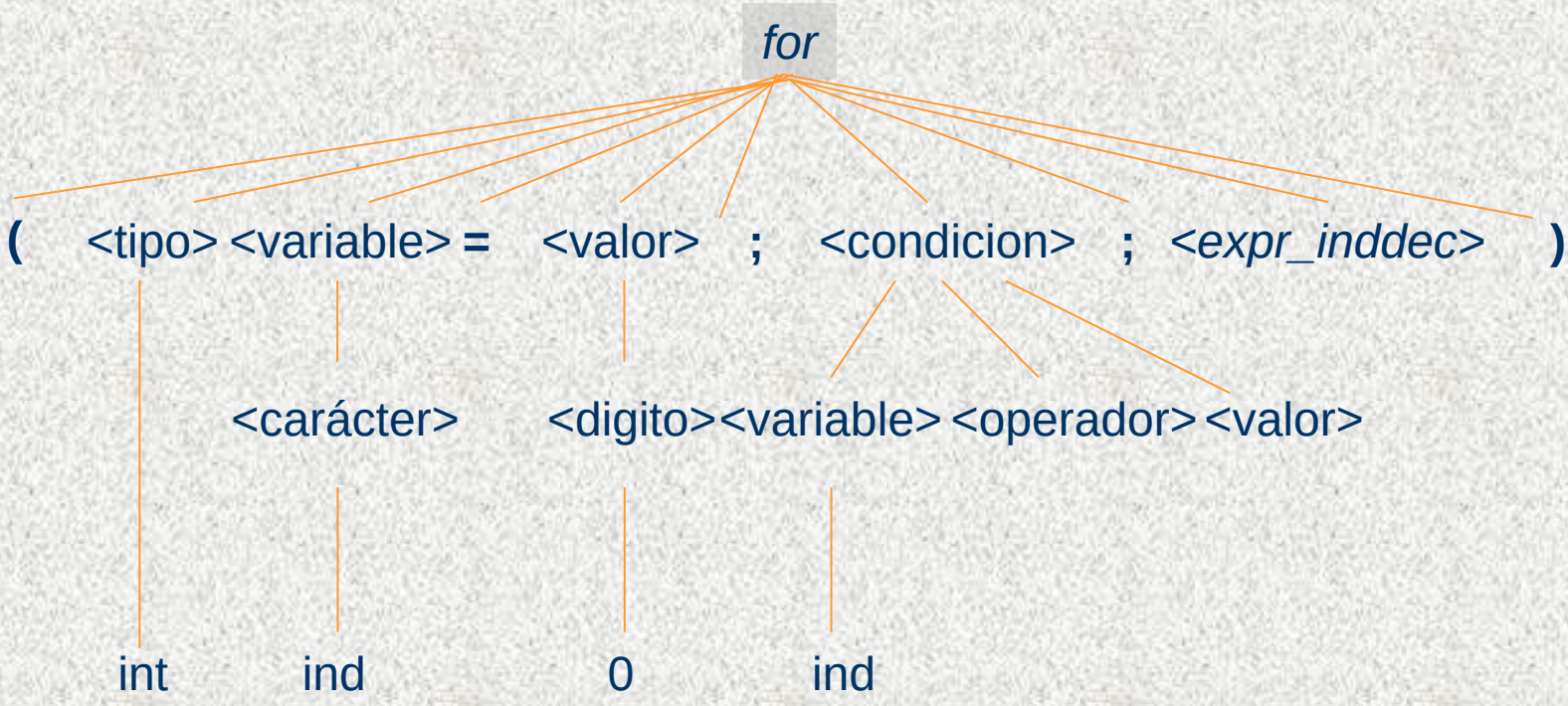
**<for>::= ‘(‘[‘,’ {<tipo> <variable> ‘=‘ <valor>}]’;’[‘,’{ <condicion>}]’;’ [‘,’{<expr_incdec>}‘)’**
**<tipo>::= ‘int’ | ‘char’ | ‘float’ | ‘double’ | ‘long’ | ‘short’ | ‘void’**
**<variable>::= {<carácter>}$^{+}$**
***<caracter>::= ‘a’* | ‘b’ |.... | ‘z’ | ‘A’ |.... | ‘Z’**
**<valor>::= {<digito>}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




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

# Ejemplo

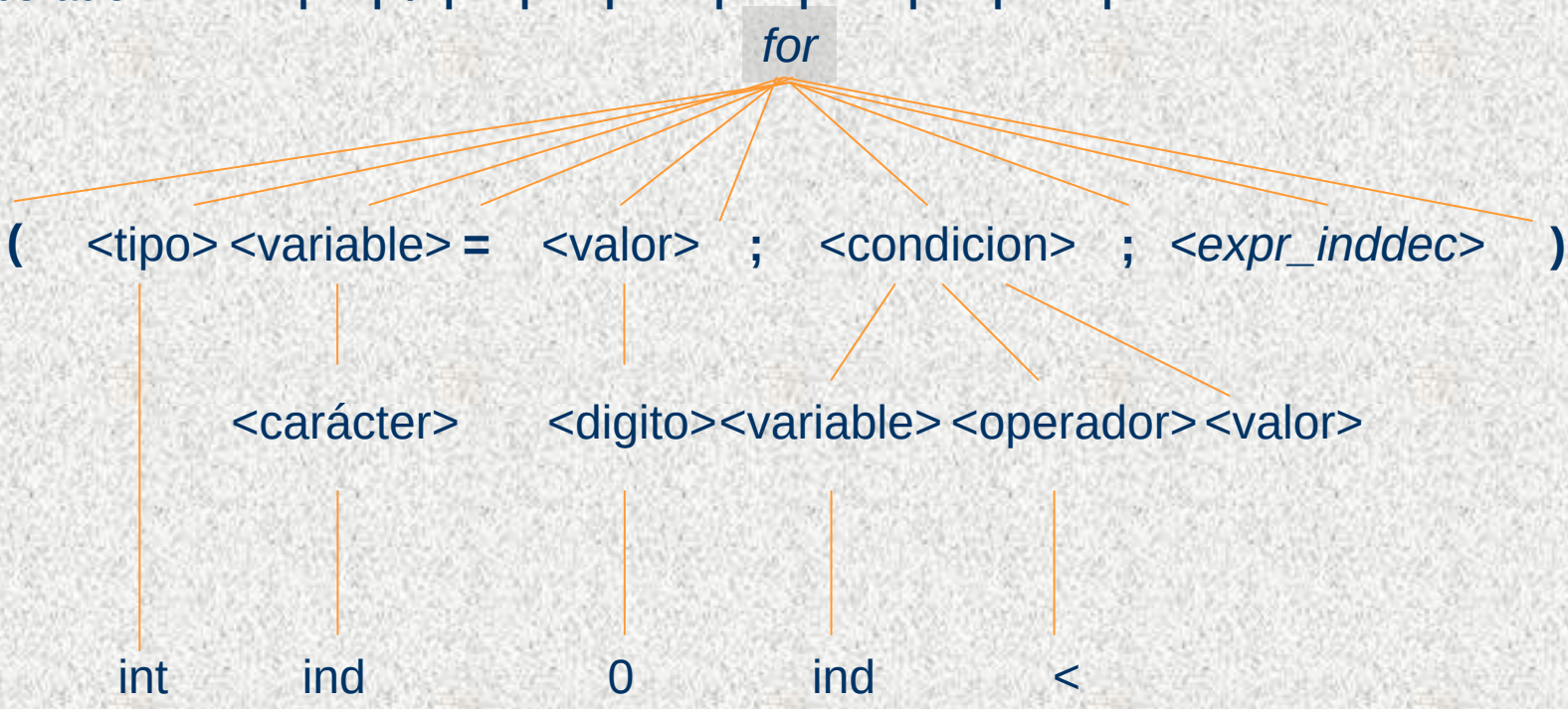
**<for>::= ‘(‘[‘,’ {<tipo> <variable> ‘=‘ <valor>}]’;’[‘,’{ <condicion>}]’;’ [‘,’{<expr_incdec>}‘)’**
**<tipo>::= ‘int’ | ‘char’ | ‘float’ | ‘double’ | ‘long’ | ‘short’ | ‘void’**
**<variable>::= {<carácter>}$^{+}$**
***<caracter>::= ‘a’* | ‘b’ |.... | ‘z’ | ‘A’ |.... | ‘Z’**
**<valor>::= {<digito>}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




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

# Ejemplo

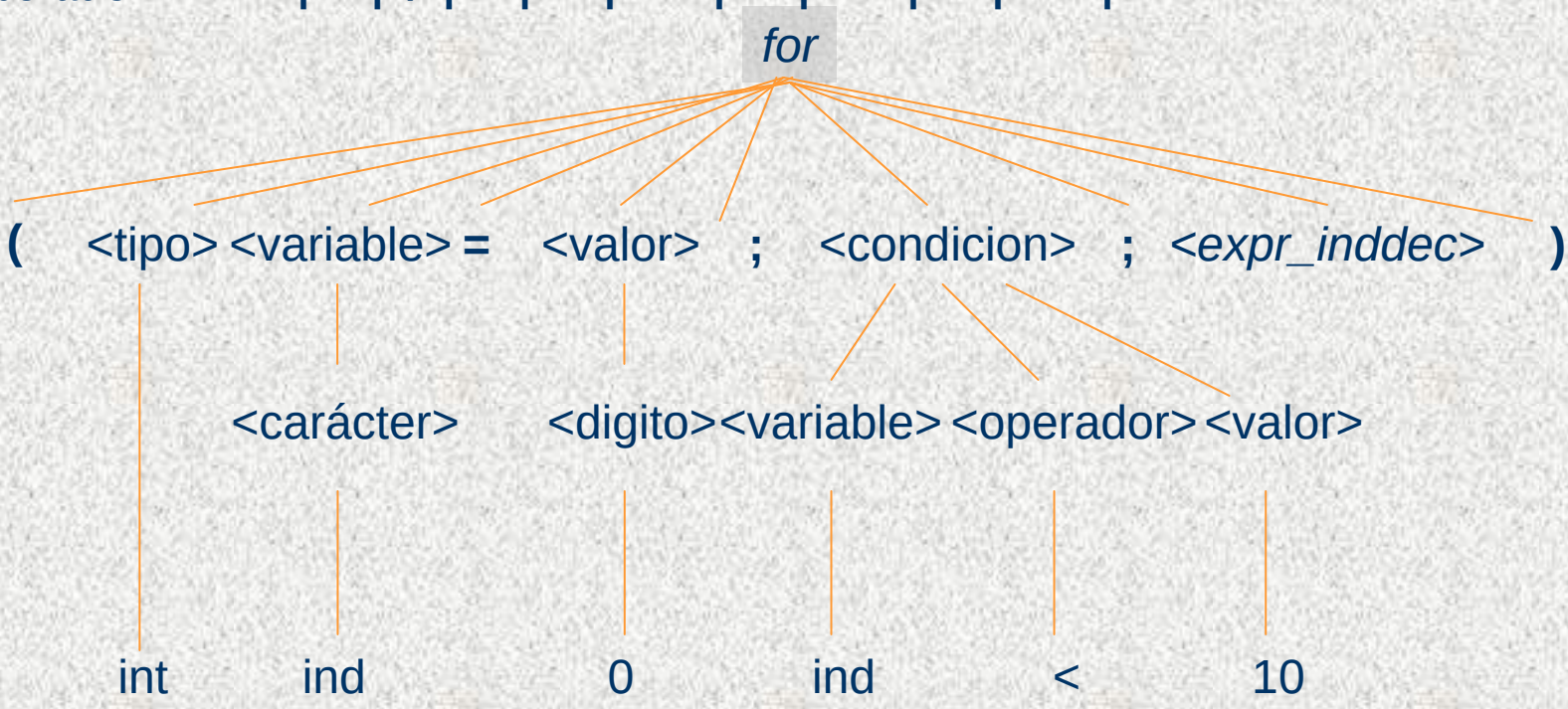
**<for>::= ‘(‘[‘,’ {<tipo> <variable> ‘=‘ <valor>}]’;’[‘,’{ <condicion>}]’;’ [‘,’{<expr_incdec>}‘)’**
**<tipo>::= ‘int’ | ‘char’ | ‘float’ | ‘double’ | ‘long’ | ‘short’ | ‘void’**
**<variable>::= {<carácter>}$^{+}$**
***<caracter>::= ‘a’* | ‘b’ |.... | ‘z’ | ‘A’ |.... | ‘Z’**
**<valor>::= {<digito>} $^{+}$**
***<digito>::= 0* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**
**<condicion>::= <variable><operador><valor>**
**<operador>::= ‘+’ | ‘-’ | ‘/’ | ‘*’ | ‘>’ | ‘>=‘ | ‘<‘ | ‘<=‘ | ‘=‘ | ‘==‘ | ‘!‘**




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

# Ejemplo

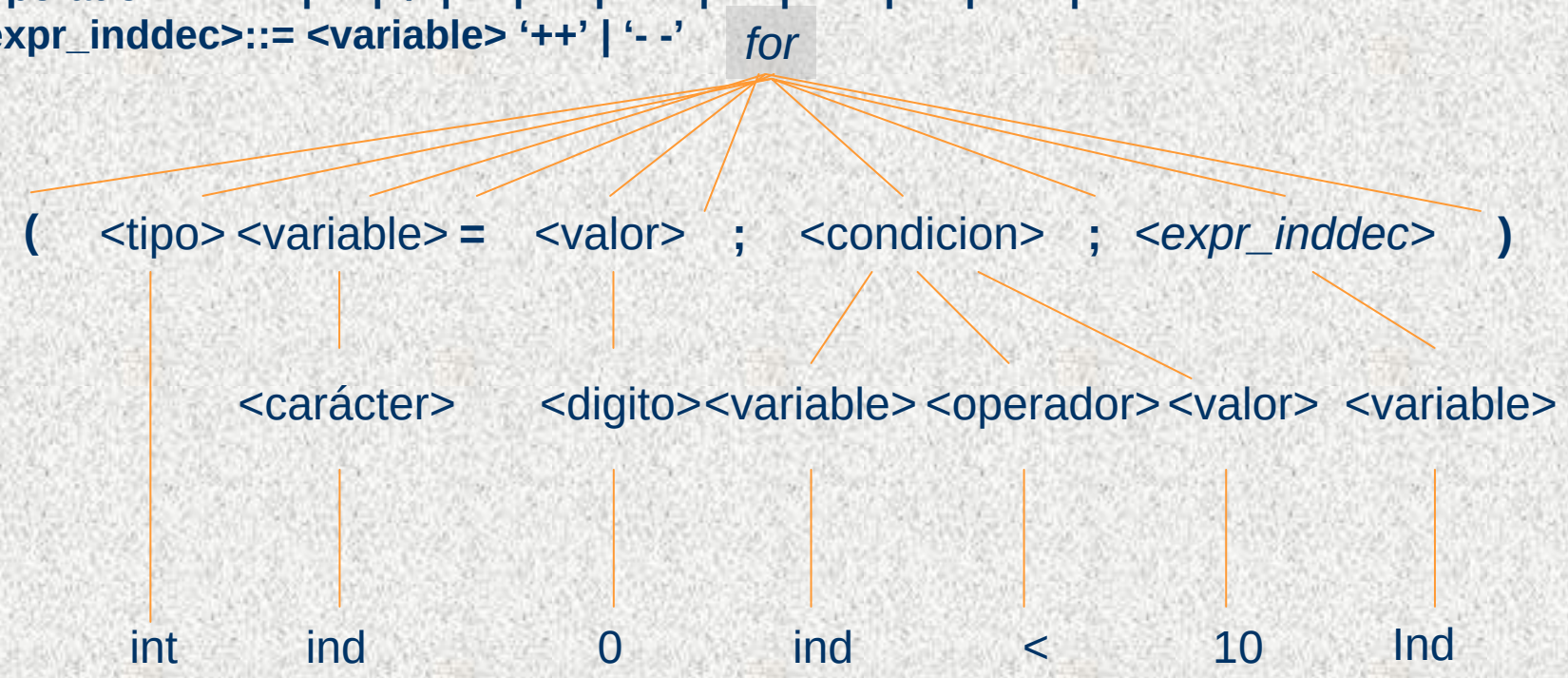
**<for>::= ‘(‘[‘,’ {<tipo> <variable> ‘=‘ <valor>}]’;’[‘,’{ <condicion>}]’;’ [‘,’{<expr_incdec>}‘)’**
**<tipo>::= ‘int’ | ‘char’ | ‘float’ | ‘double’ | ‘long’ | ‘short’ | ‘void’**
**<variable>::= {<carácter>}$^{+}$**
***<caracter>::= ‘a’* | ‘b’ |.... | ‘z’ | ‘A’ |.... | ‘Z’**
**<valor>::= {<digito>} $^{+}$**
***<digito>::= 0* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**
**<condicion>::= <variable><operador><valor>**
**<operador>::= ‘+’ | ‘-’ | ‘/’ | ‘*’ | ‘>’ | ‘>=‘ | ‘<‘ | ‘<=‘ | ‘=‘ | ‘==‘ | ‘!‘**




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

# Ejemplo

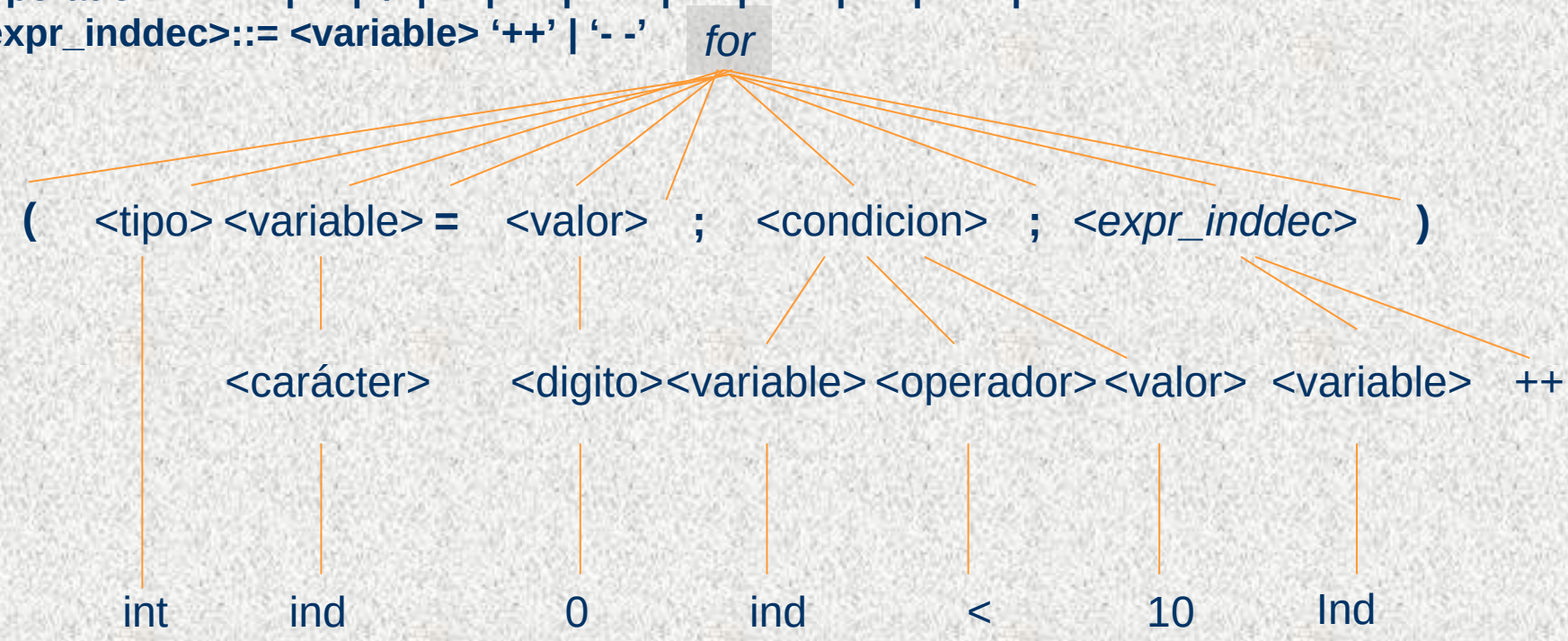
**<for>::= ‘(‘[‘,’ {<tipo> <variable> ‘=‘ <valor>}]’;’[‘,’{ <condicion>}]’;’ [‘,’{<expr_incdec>}‘)’**
**<tipo>::= ‘int’ | ‘char’ | ‘float’ | ‘double’ | ‘long’ | ‘short’ | ‘void’**
**<variable>::= {<carácter>}$^{+}$**
***<caracter>::= ‘a’* | ‘b’ |.... | ‘z’ | ‘A’ |.... | ‘Z’**
**<valor>::= {<digito>} $^{+}$**
***<digito>::= 0* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**
**<condicion>::= <variable><operador><valor>**
**<operador>::= ‘+’ | ‘-’ | ‘/’ | ‘*’ | ‘>’ | ‘>=‘ | ‘<‘ | ‘<=‘ | ‘=‘ | ‘==‘ | ‘!‘**
**<expr_inddec>::= <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>::= {<digito>} +**
***<digito>::= 0* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**
**<condicion>::= <variable><operador><valor>**
**<operador>::= ‘+’ | ‘-’ | ‘/’ | ‘*’ | ‘>’ | ‘>=‘ | ‘<‘ | ‘<=‘ | ‘=‘ | ‘==‘ | ‘!‘**
**<expr_inddec>::= <variable> ‘++’ | ‘- -’**




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