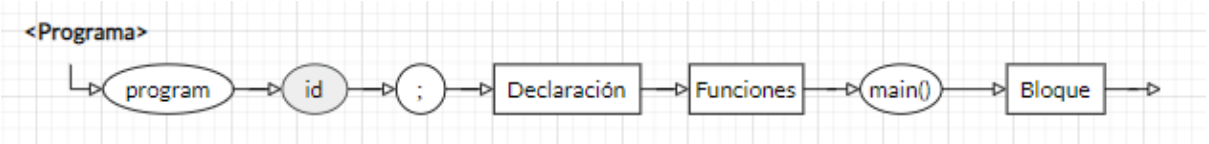
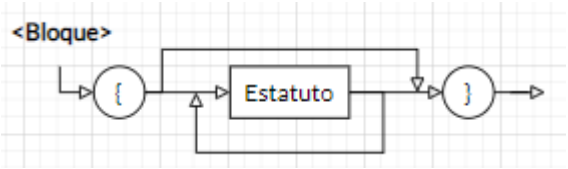
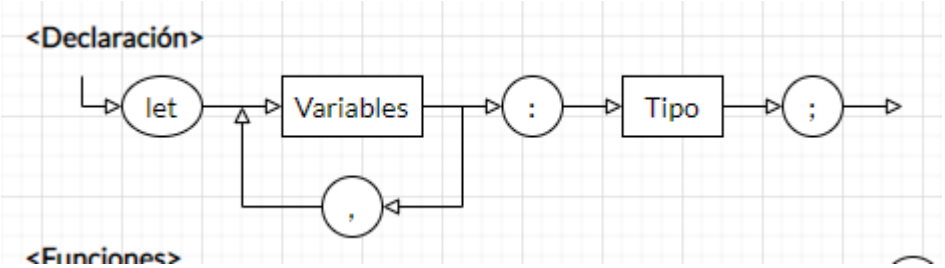


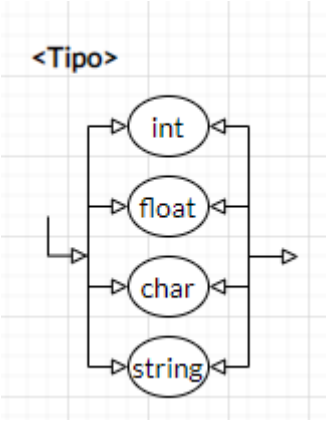
Programa → program <id> ; <Declaración> <Funciones> main() <Bloque>



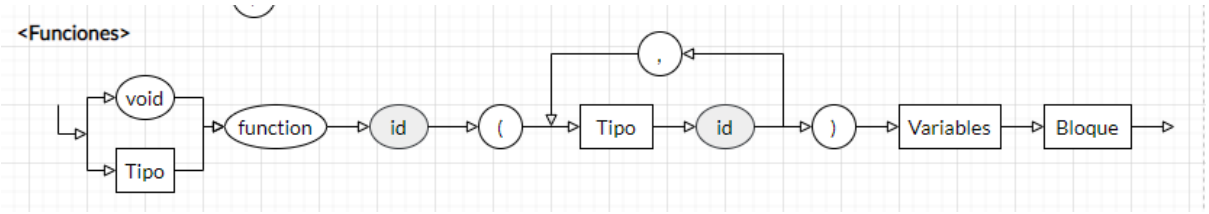
Bloque → {<Bloque_prime>}
Bloque_prime → <Estatuto>; <Bloque_prime> | eps



Declaración → <Declaración_base> | <Declaración_base> <Declaración>
Declaración_base → let <Declaración_prime> : <Tipo>;
Declaración_prime → <Variable> | <Variable>, <Declaración_prime>

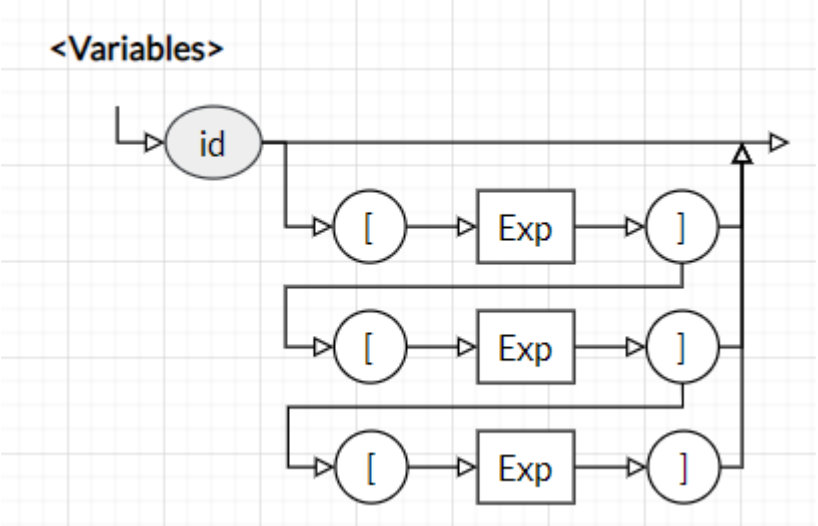


Tipo → int | float | char | string

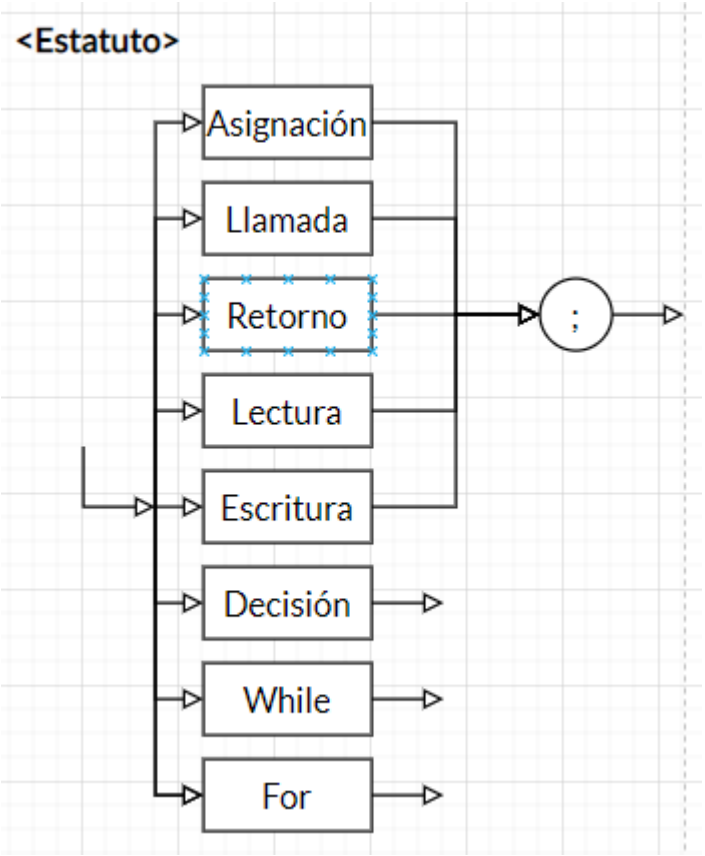


Funciones → <Funcion_base> | <Funcion_base> <Funciones>

Funciones_base → function <Func_Type> <id> (<Funciones_prime>) <Declaración>
<Bloque>
Funciones_prime → <Tipo> <id> | <Tipo> <id> , <Funciones_prime>
Func_type → void | <Tipo>

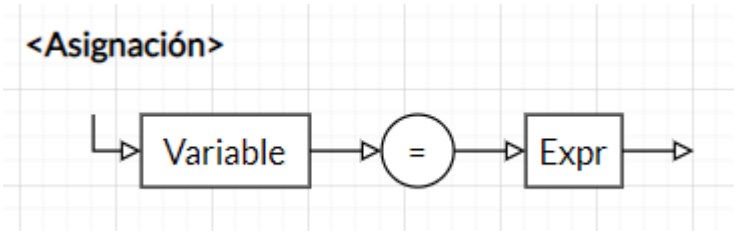


Variable → <id>
| <id> [<Exp>]
| <id> [<Exp>][<Exp>]
| <id> [<Exp>][<Exp>][<Exp>]

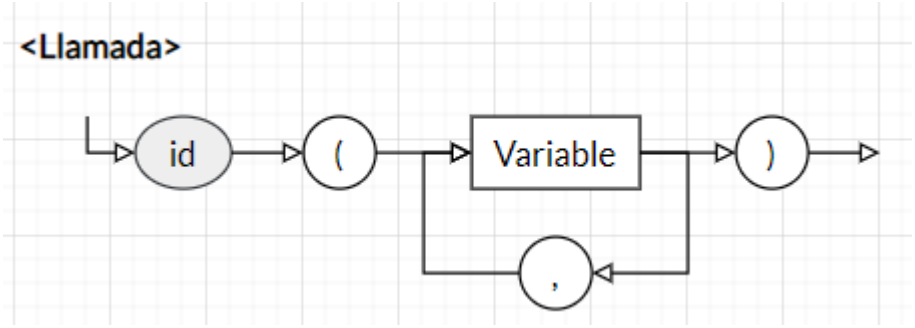


Estatuto → <Asignación> ;

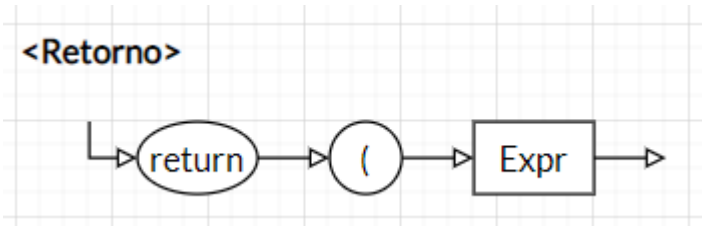
| <Llamada> ;
| <Retorno> ;
| <Lectura> ;
| <Escritura> ;
| <Decisión>
| <While>
| <For>
| <Expr>



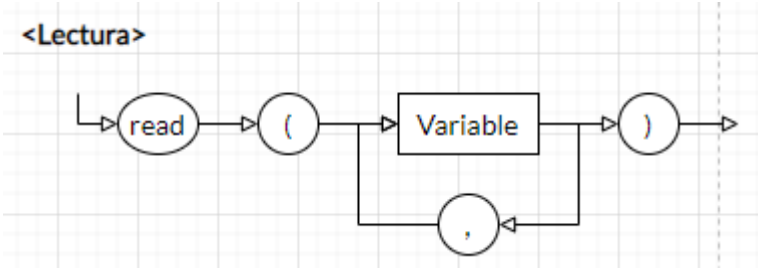
Asignación → <Variable> = <Expr>



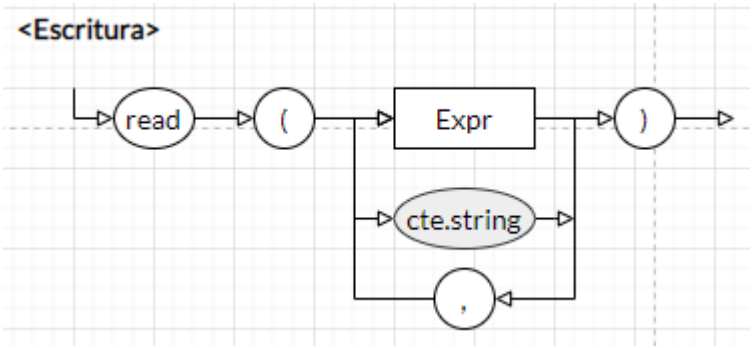
Llamada → <id> (<Llamada_prime>)
<Llamada_prime> → <Exp> | <Exp>, <Llamada_prime>



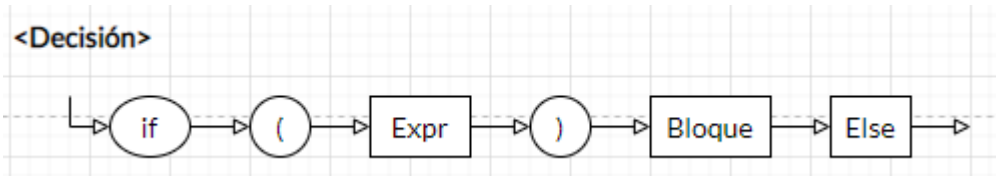
Retorno → return (<Expr>)



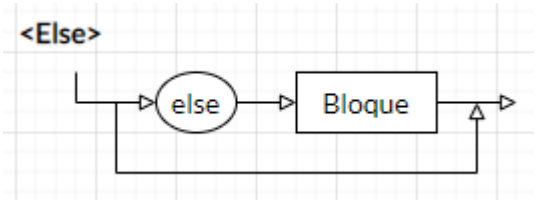
Lectura → read (<Lectura_prime>)
Lectura_prime → → <Variable> | <Variable>, <Lectura_prime>



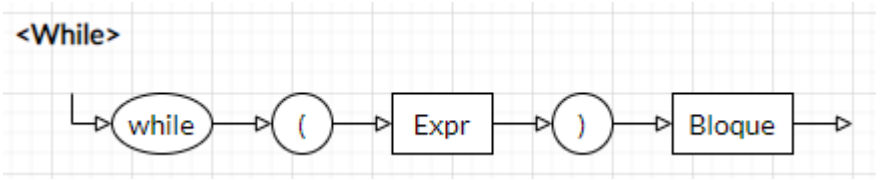
Escritura → write (<Escritura_prime>)
Escritura_prime → <Expr>
| <Cte_string>
| <Expr>, <Escritura_prime>
| <Cte_string>, <Escritura_prime>



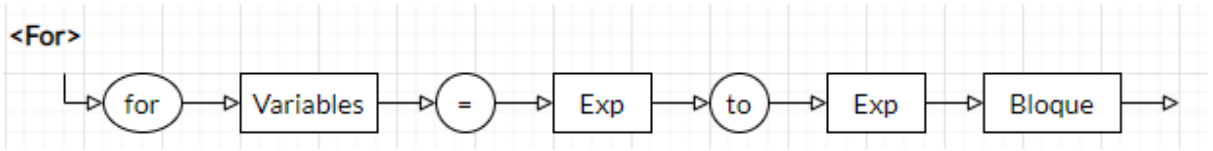
Decisión → if (<Expr>) <Bloque> <Else>



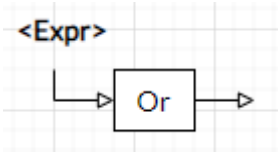
Else → else <Bloque> | eps



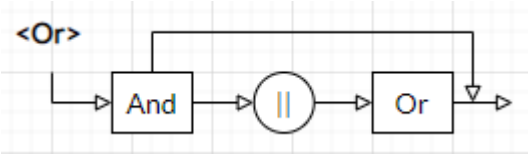
While → while (<Expr>) <Bloque>



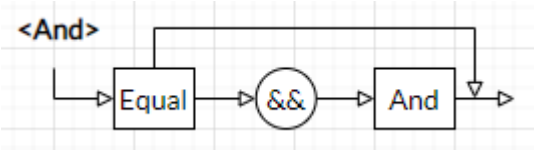
For → for <Variable> = <Exp> to <Exp> <Bloque>



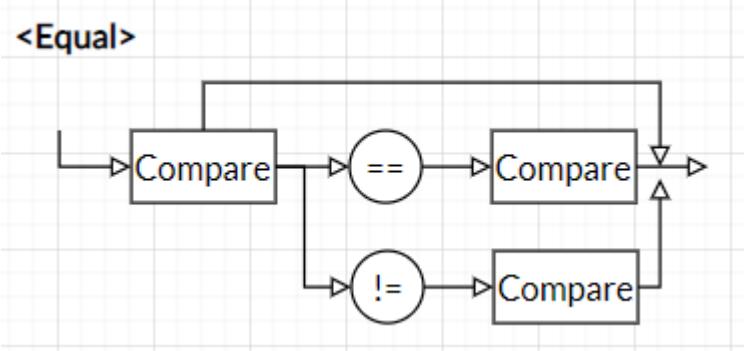
Expr -> <Or>



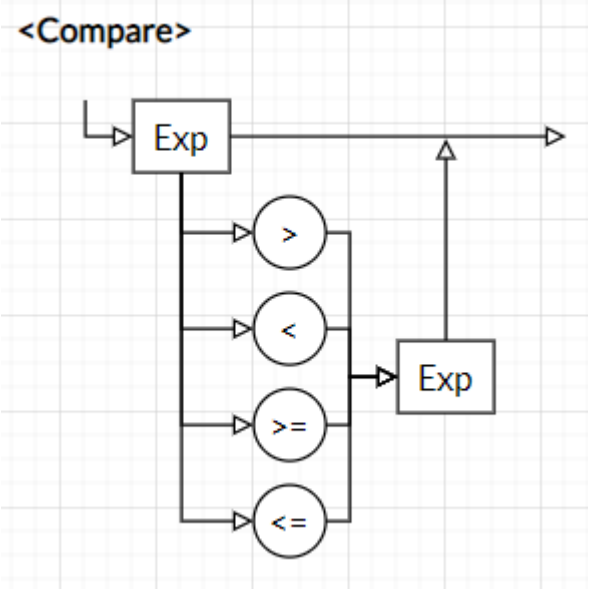
Or → <And>
| <And> || <Or>



And → <Equal>
| <Equal> && <And>

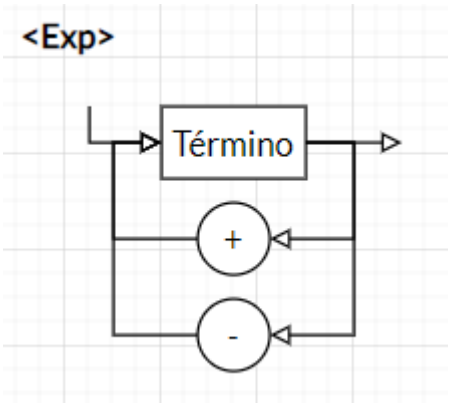


Equal → <Compare>
| <Compare> == <Compare>
| <Compare> != <Compare>

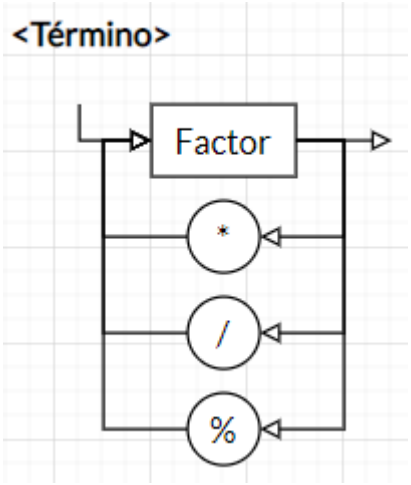


Compare → <Exp>
| <Exp> > <Exp>
| <Exp> < <Exp>
| <Exp> >= <Exp>

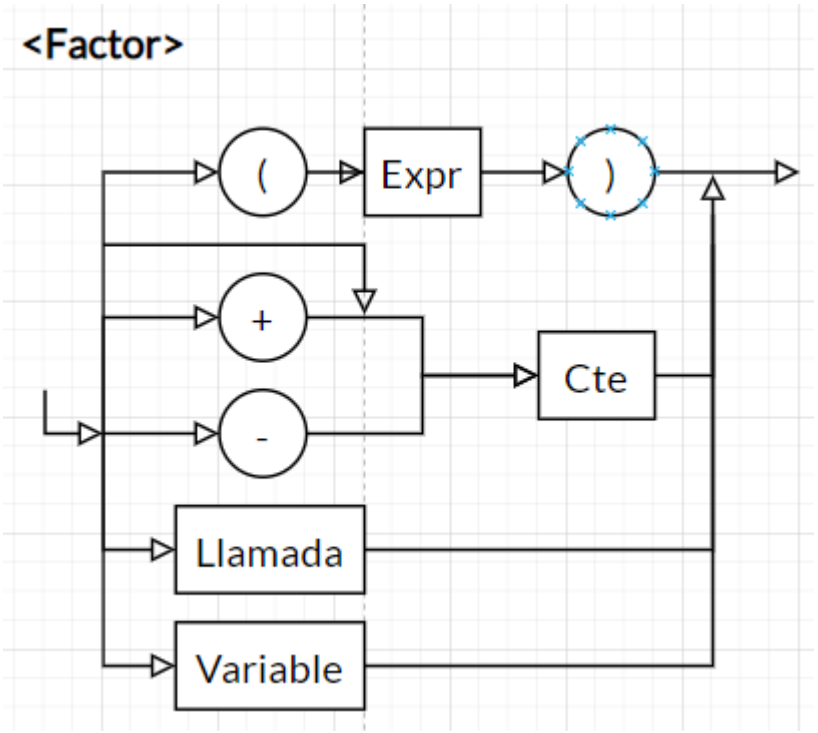
| <Exp> <= <Exp>



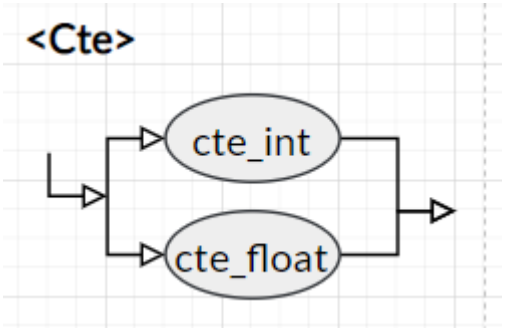
Exp → <Termino>
| <Termino> + <Exp>
| <Termino> - <Exp>



Termino → <Factor>
| <Factor> * <Termino>
| <Factor> / <Termino>
| <Factor> % <Termino>



Factor → <Variable>
| (<Expr>)
| <Llamada>
| <Cte>
| + <Cte>
| - <Cte>



<Cte> → <Cte_int> | <Cte_float>

```
tokens = [
    # palabras reservadas
    'PROGRAM',
    'MAIN',
    'LET',
    'INT',
    'FLOAT',
    'CHAR',
    'STRING',
    'FUNCTION',
```

```
    'VOID',
    'RETURN',
    'READ',
    'WRITE',
    'IF',
    'ELSE',
    'WHILE',
    'FOR',
    'TO',
    # puntuacion
    'O_CBRACKET',
    'C_CBRACKET',
    'O_PARENTHESIS',
    'C_PARENTHESIS',
    'O_ABRACKET',
    'C_ABRACKET',
    'SEMICOLON',
    'COLON',
    'COMMA',
    # operadores
    'ASSIGN',
    'OR',
    'AND',
    'EQUAL',
    'NOT_EQUAL',
    'GREATER',
    'LESSER',
    'GREATER_EQUAL',
    'LESSER_EQUAL',
    'PLUS',
    'MINUS',
    'TIMES',
    'DIVIDE',
    'MODULE',
    # regex
    'ID',
    'CTE_INT',
    'CTE_FLOAT',
    'CTE_CHAR',
    'CTE_STRING'
]
```