PyCLite ?

Bases

El nombre del lenguaje es porque la sintaxis del lenguaje está basada en algo sencillo como Python con cierta inspiración en c++ , de modo que no sea complicado entender las instrucciones creadas.

Palabras reservadas

- char
- int
- float
- bool
- array
- If
- for
- while
- func
- csay
- cread

Tipos de datos

- Cadena de caracteres (char)
- Enteros (int)
- Decimales (float)
- Booleanos (bool)
- Arreglos (array)

Símbolos

- Operadores
    - \+ (suma)
    - – (resta)

- * (multiplicacion)
- / (division)
- % (modulo)
- == (igualdad)
- != (desigualdad)
- > (mayor que)
- < (menor que)
- <= (igual o mayor que)
- >= (igual o menor que)
- && (and)
- || (or)

Sintaxis - ejemplos

- Cometarios se harán usando “//” o “$” para comentario de línea y “/**/” o “%%” para comentarios de bloque
  - // Este sería un comentario
  - % Este sería un
    comentario de bloque %

- Se asignan valores con “=”, las variables son char, int, float, bool
  - int x = 10;
  - char name = "Rodrigo";
  - bool active = true;

- If
  - if (<condition>) {
    # code
    }

- For
  - for (i in x) {
    # code
    }

- While
    - while (<condition>) {
      # code
      }

- Funciones
    - func name(param1, param2) {
      # code
      return <something>;
      }

- Array
    - array nums = [1, 2, 3, 4];

- Los outputs serán usando la palabra “csay”
    - csay ("Hello World");

- Los inputs serán usando la palabra “cread” y el valor se almacenará en la variable especificada a continuación
    - cread ("Enter value: ") x;

Gramática (de la forma LL1)

PROGRAMA

```
<programa> ::= <lista_instrucciones>
<lista_instrucciones> ::= <instrucción> <lista_instrucciones> | ε
```

INSTRUCCIÓN

```
<instrucción> ::= <declaración_variable> ";"
                | <asignación> ";"
                | <if>
                | <for>
                | <while>
```

```
                | <función>
                | <llamada_función> ";"
                | <comentario>
                | <return>
                | <primario> ";"
```

DECLARACIÓN DE VARIABLE

```
 <declaración_variable> ::= <tipo> <identificador> "=" <expresión>
                        | "array" <identificador> "=" "[" <lista_valores> "]"
 <tipo> ::= "int" | "float" | "char" | "bool"
```

ASIGNACIÓN

```
 <asignación> ::= <identificador> "=" <expresión>
```

EXPRESIÓN

```
<expresión> ::= <or_expr>

<or_expr> ::= <and_expr> <or_expr'>
      <or_expr'> ::= "||" <and_expr> <or_expr'> | ε

<and_expr> ::= <eq_expr> <and_expr'>
      <and_expr'> ::= "&&" <eq_expr> <and_expr'> | ε

<eq_expr> ::= <rel_expr> <eq_expr'>
      <eq_expr'> ::= ("==" | "!=") <rel_expr> <eq_expr'> | ε

<rel_expr> ::= <add_expr> <rel_expr'>
      <rel_expr'> ::= (">" | "<" | ">=" | "<=") <add_expr> <rel_expr'> | ε

<add_expr> ::= <mul_expr> <add_expr'>
      <add_expr'> ::= ("+" | "-") <mul_expr> <add_expr'> | ε

<mul_expr> ::= <unary_expr> <mul_expr'>
      <mul_expr'> ::= ("*" | "/" | "%") <unary_expr> <mul_expr'> | ε

<unary_expr> ::= ("-" | "!" | "++" | "--") <unary_expr> | <primario>

<primario> ::= <valor>
| <identificador> <primario_tail>
| "(" <expresión> ")"
      <primario_tail> ::= "(" <argumentos_opt> ")" | ε
```

CONDICIONAL IF

```
 <if> ::= "if" "(" <expresión> ")" "{" <lista_instrucciones> "}"
```

BUCLE FOR

```
 <for> ::= "for" "(" <identificador> "in" <identificador> ")" "{"
<lista_instrucciones> "}"
```

BUCLE WHILE

```
 <while> ::= "while" "(" <expresión> ")" "{" <lista_instrucciones> "}"
```

FUNCIONES

```
 <función> ::= "func" <identificador> "(" <parametros_opt> ")" "{"
      <lista_instrucciones> <return_opt> "}"
 <parametros_opt> ::= <parametros> | ε
 <parametros> ::= <identificador> <parametros'>
 <parametros'> ::= "," <identificador> <parametros'> | ε

<return_opt> ::= <return> | ε
<return> ::= "return" <expresión> ";"
```

LLAMADAS ESPECIALES

```
 <llamada_función> ::= "csay" "(" <expresión> ")"
                     | "cread" "(" <expresión> ")" <identificador>
                     | <identificador> "(" <argumentos_opt> ")"

<argumentos_opt> ::= <argumentos> | ε
      <argumentos> ::= <expresión> <argumentos'>
      <argumentos'> ::= "," <expresión> <argumentos'> | ε
```

COMENTARIOS

```
 <comentario> ::= "//" <texto>      comentario línea
               | "$" <texto>        comentario línea alternativo
               | "%%" <texto> "%%"  comentario bloque
               | "/" <texto> "/"    comentario bloque alternativo
```

VALORES Y LISTAS

```
 <valor> ::= <número> | <cadena> | "true" | "false"
 <lista_valores> ::= <valor> <lista_valores'>
 <lista_valores'> ::= "," <valor> <lista_valores'> | ε
 <número> ::= [ "-" ] <dígito> { <dígito> } [ "." <dígito> { <dígito> } ]
      <dígito> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
 <cadena> ::= '"' { cualquier_caracter } '"'
 <identificador> ::= letra { letra | dígito | "_" }
      <letra> ::= "a" | ... | "z" | "A" | ... | "Z"
```

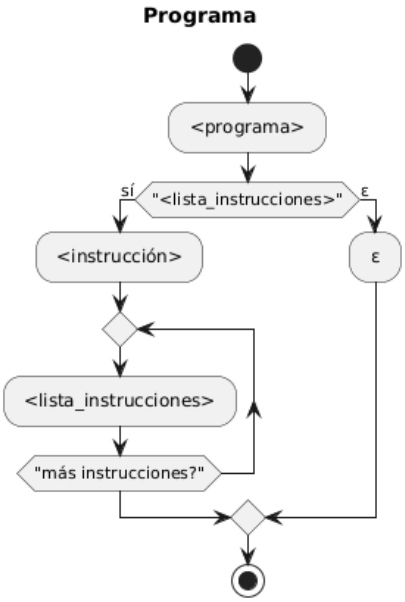
Notación utilizada:

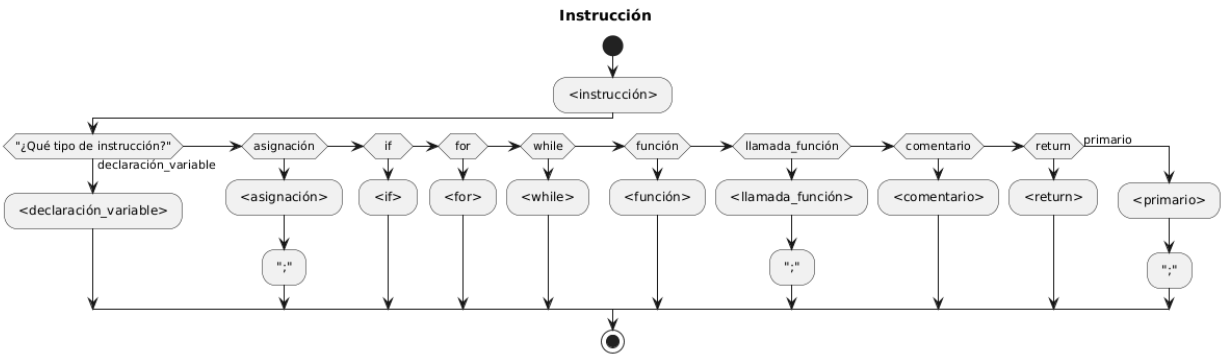
Símbolo	Significado
<...>	No terminal (categoría gramatical)
"..."	Terminal literal (palabra reservada, símbolo)

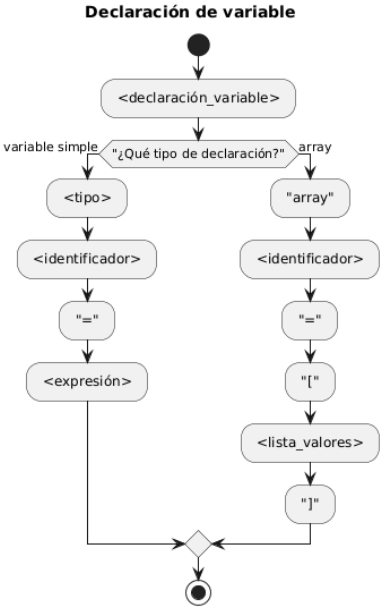
::=	Se define como
|	Alternativa (OR)
{ ... }	Repetición: cero o más veces
[...]	Opcional: una vez o ninguna
ε	Cadena vacía
(...)	Agrupación literal (paréntesis reales en código)

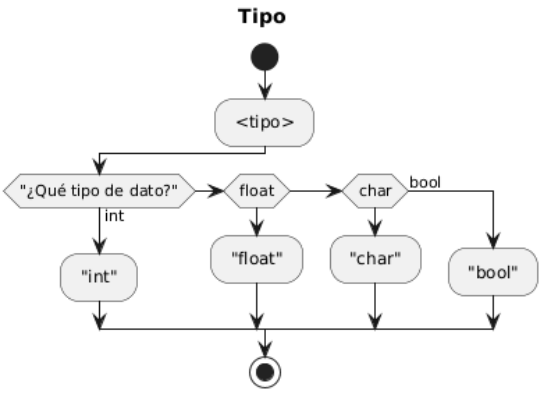
Palabras reservadas: **int**, **float**, **char**, **bool**, **array**, **if**, **for**, **in**, **while** **func**, **return**, **true**, **false**, **csay**, **cread**

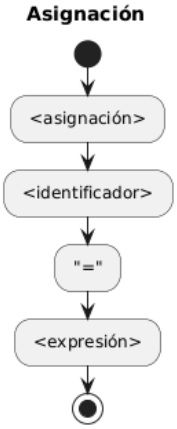
Diagramas de sintaxis

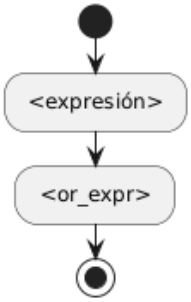


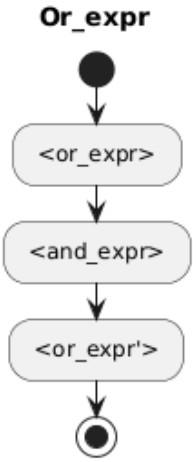


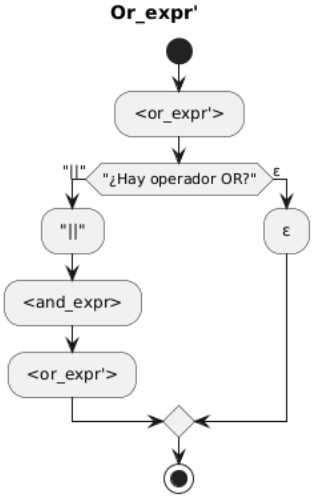

Declaración de variable
<declaración_variable>
"¿Qué tipo de declaración?"
variable simple
array
<tipo>
<identificador>
"="
<expresión>
"array"
<identificador>
"="
"["
<lista_valores>
"]"

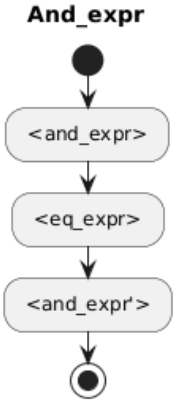

Tipo
<tipo>
"¿Qué tipo de dato?"
int
float
char
bool
"int"
"float"
"char"
"bool"

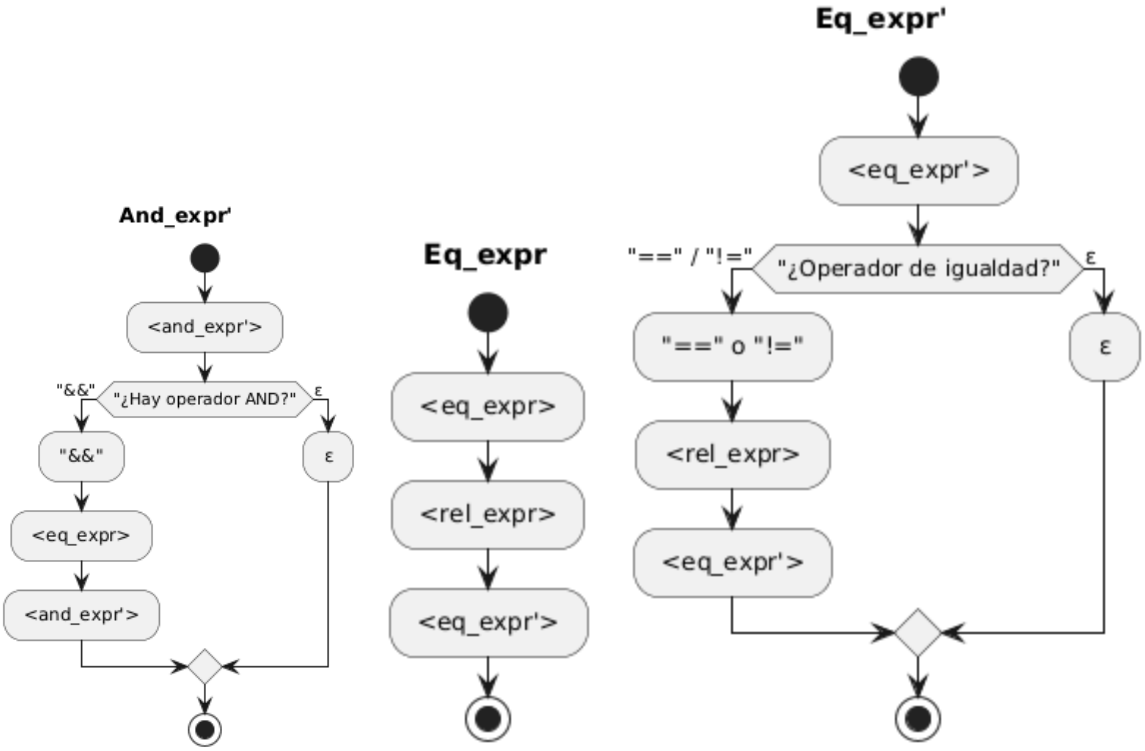

Asignación
<asignación>
<identificador>
"="
<expresión>

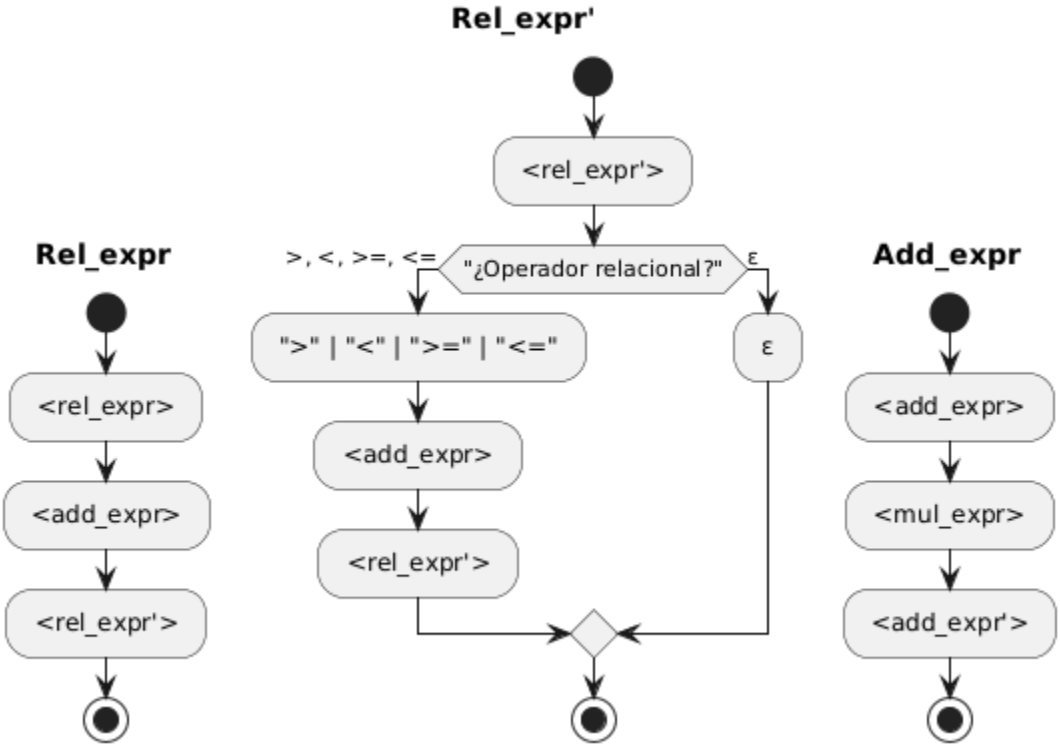

Expresión
<expresión>
<or_expr>

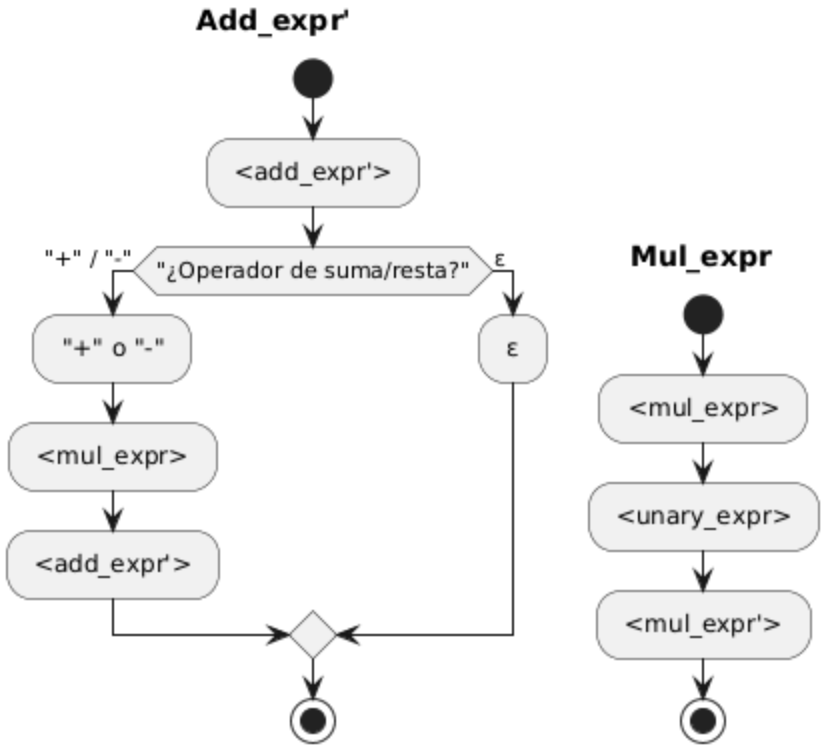

Or_expr
<or_expr>
<and_expr>
<or_expr'>

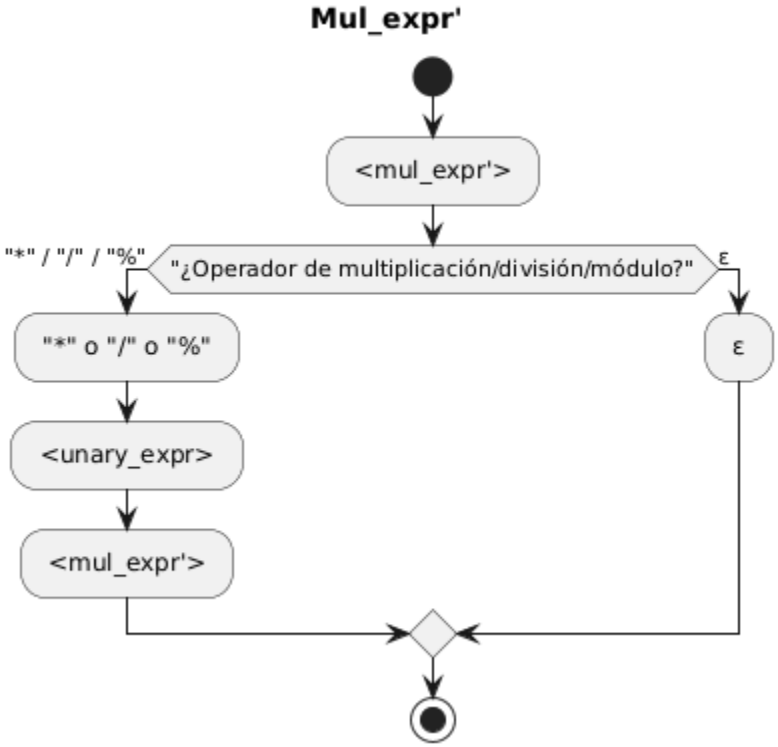

Or_expr'
<or_expr'>
"¿Hay operador OR?"
"||"
ε
"||"
<and_expr>
<or_expr'>

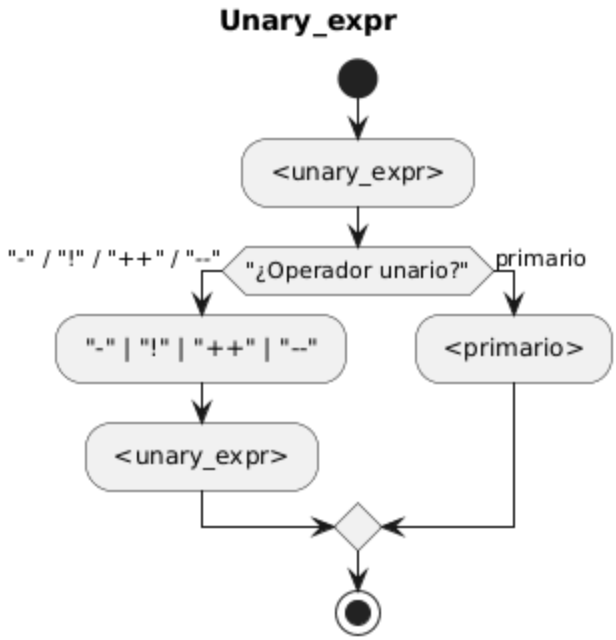

And_expr
<and_expr>
<eq_expr>
<and_expr'>

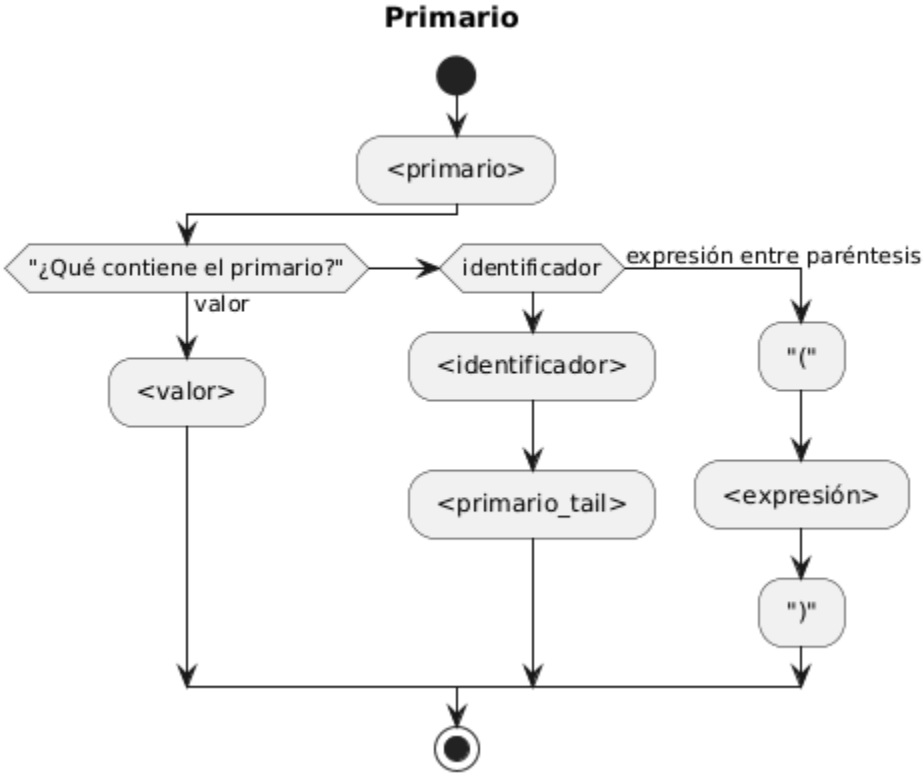

Eq_expr'
<eq_expr'>
"==" / "!="
"¿Operador de igualdad?"
ε
"==" o "!="
ε
<rel_expr>
<eq_expr'>
And_expr'
<and_expr'>
"&&"
"¿Hay operador AND?"
ε
"&&"
ε
<eq_expr>
<and_expr'>
Eq_expr
<eq_expr>
<rel_expr>
<eq_expr'>

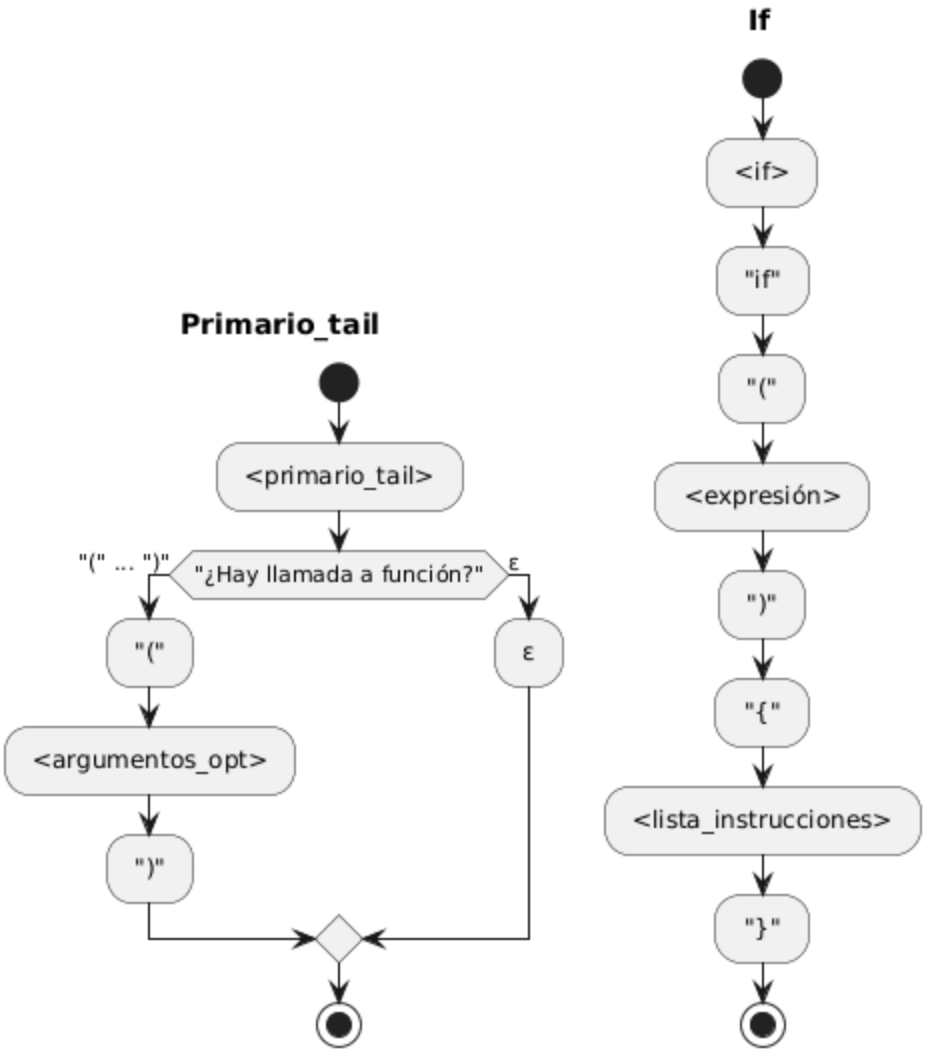

Rel_expr'
<rel_expr'>
>, <, >=, <=
"¿Operador relacional?"
ε
">" | "<" | ">=" | "<="
ε
<add_expr>
<rel_expr'>
Rel_expr
<rel_expr>
<add_expr>
<rel_expr'>
Add_expr
<add_expr>
<mul_expr>
<add_expr'>


Add_expr'
<add_expr'>
"¿Operador de suma/resta?"
"+" / "-"
ε
"+" o "-"
ε
<mul_expr>
<add_expr'>
Mul_expr
<mul_expr>
<unary_expr>
<mul_expr'>

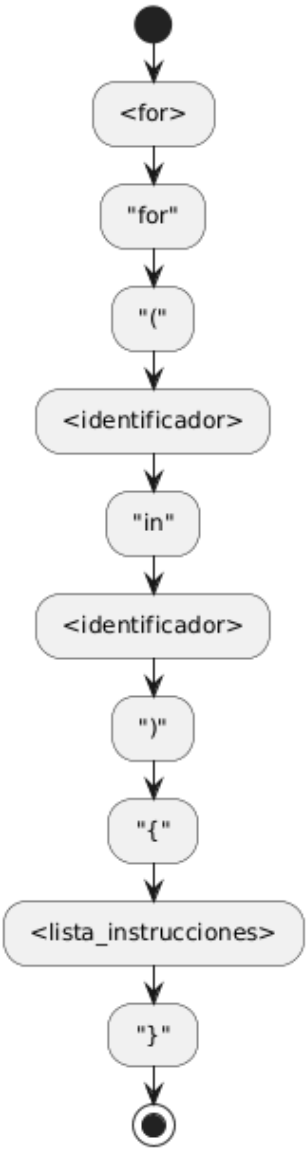

Mul_expr'
<mul_expr'>
"¿Operador de multiplicación/división/módulo?"
"*" / "/" / "%"
ε
"*" o "/" o "%"
<unary_expr>
<mul_expr'>


Unary_expr
<unary_expr>
"-" / "!" / "++" / "--"
"¿Operador unario?"
primario
"-" | "!" | "++" | "--"
<primario>
<unary_expr>

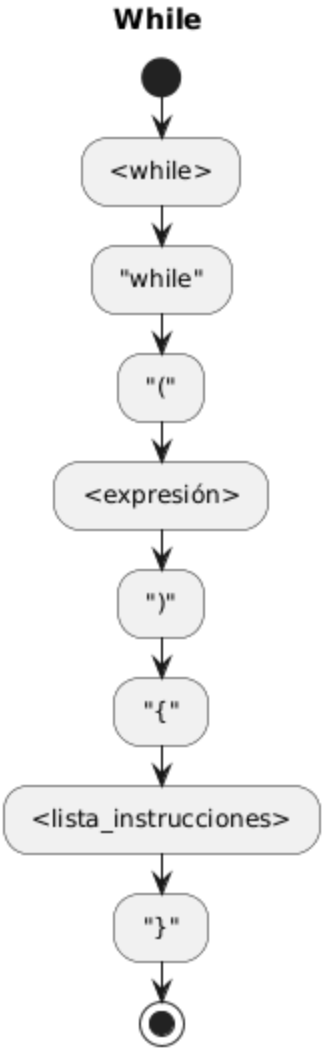

Primario
<primario>
"¿Qué contiene el primario?"
identificador
expresión entre paréntesis
valor
<valor>
<identificador>
"("
<primario_tail>
<expresión>
")"


Primario_tail
<primario_tail>
"¿Hay llamada a función?"
"(" ... ")"
ε
"("
<argumentos_opt>
")"
ε
If
<if>
"if"
"("
<expresión>
")"
"{"
<lista_instrucciones>
"}"

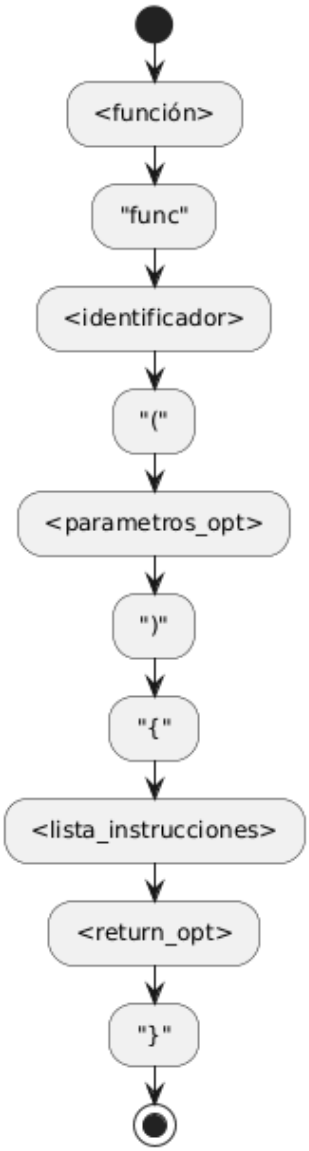
For

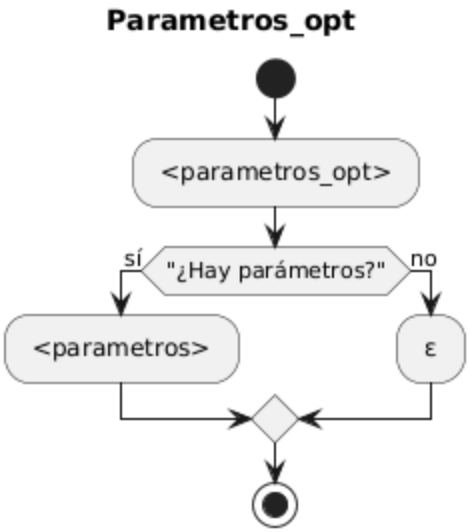


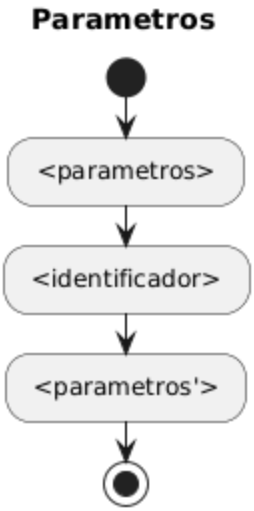
While

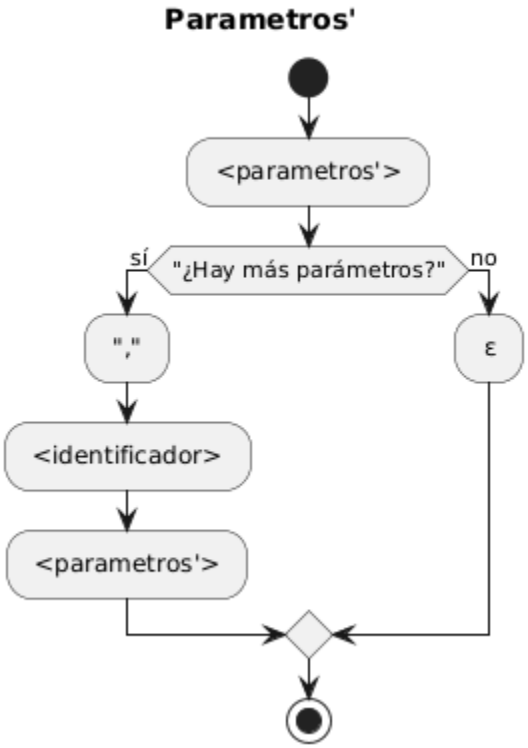


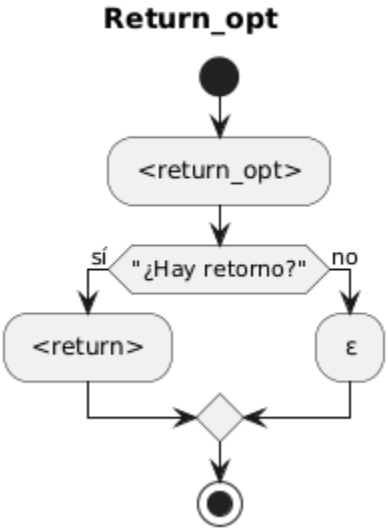
Función

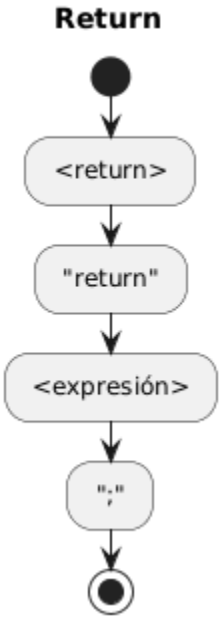


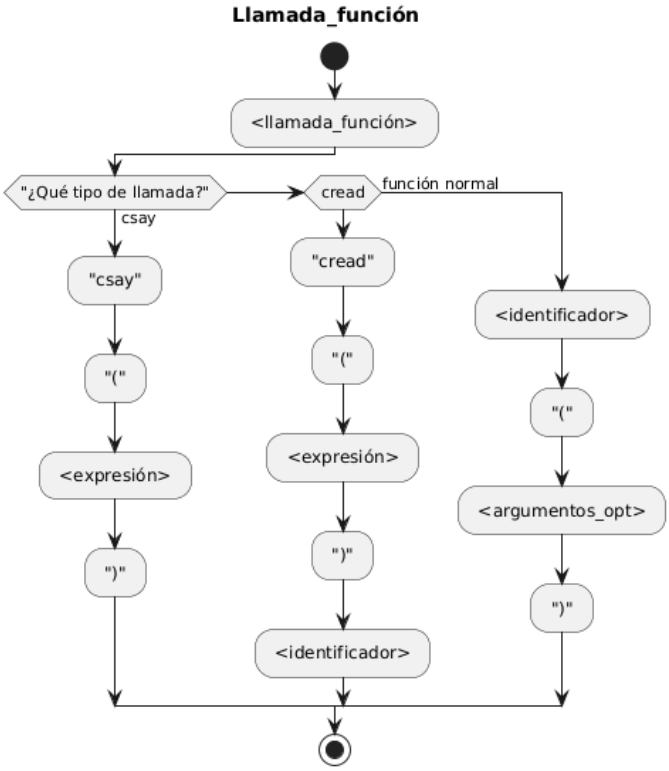

Parametros_opt
<parametros_opt>
"¿Hay parámetros?"
sí
no
<parametros>
ε

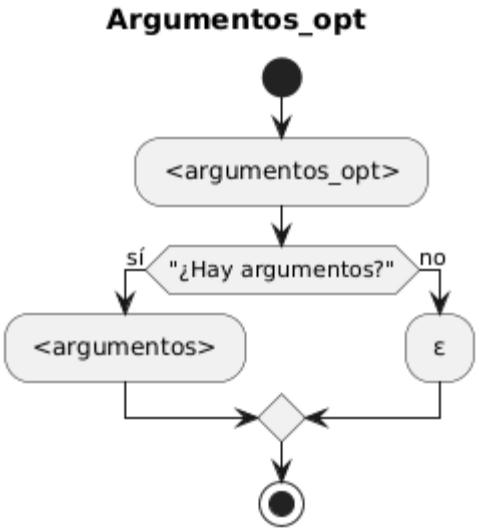

Parametros
<parametros>
<identificador>
<parametros'>

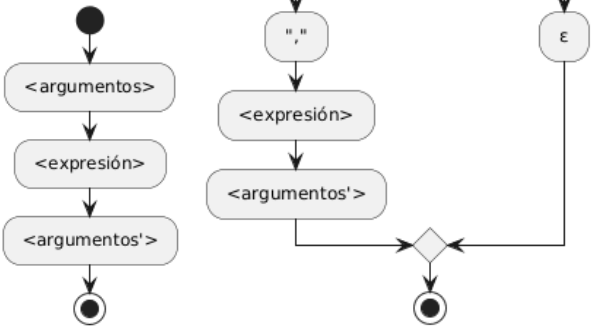

Parametros'
<parametros'>
"¿Hay más parámetros?"
sí
no
","
ε
<identificador>
<parametros'>


Return_opt
<return_opt>
"¿Hay retorno?"
sí
no
<return>
ε

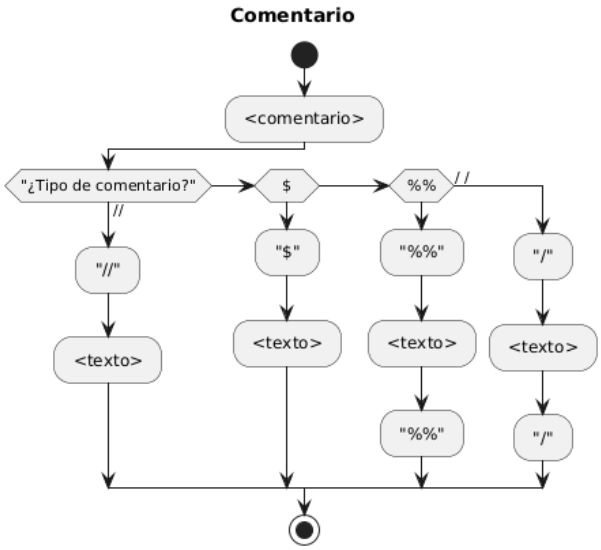

Return
<return>
"return"
<expresión>
";"

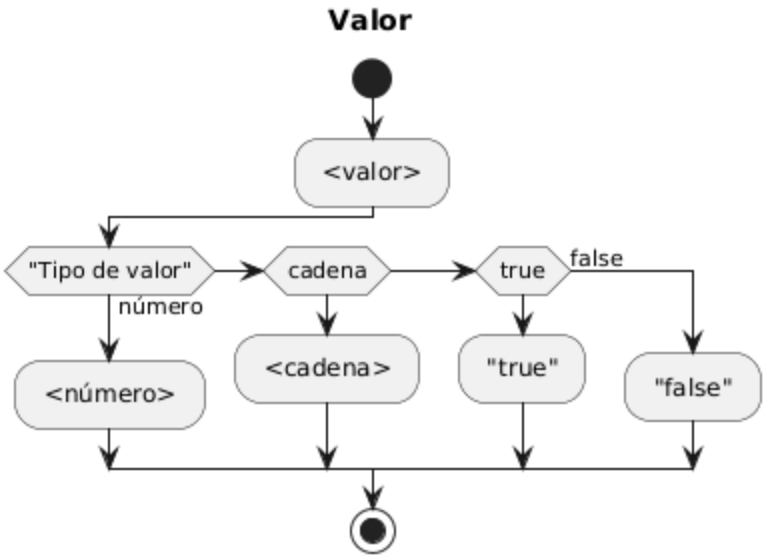

Llamada_función
<llamada_función>
"¿Qué tipo de llamada?"
csay
cread
función normal
"csay"
"("
<expresión>
")"
"cread"
"("
<expresión>
")"
<identificador>
<identificador>
"("
<argumentos_opt>
")"

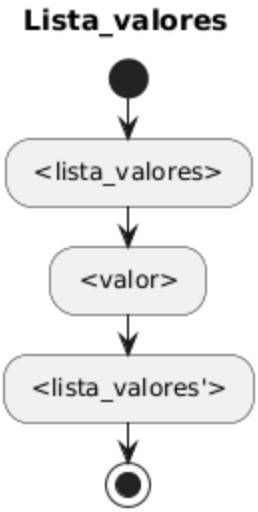

Argumentos_opt
<argumentos_opt>
sí
"¿Hay argumentos?"
no
<argumentos>
ε

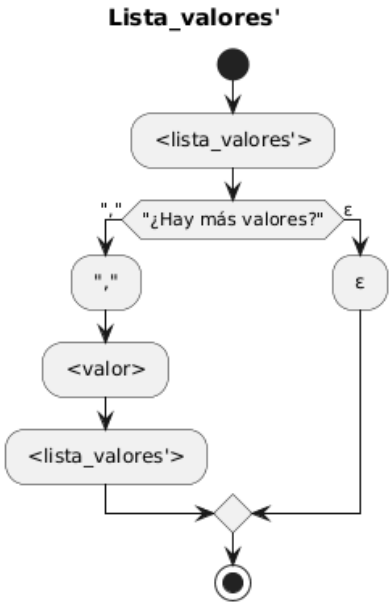

Argumentos
<argumentos>
<expresión>
<argumentos'>

Argumentos'
<argumentos'>
","
"¿Hay más argumentos?"
ε
","
ε
<expresión>
<argumentos'>

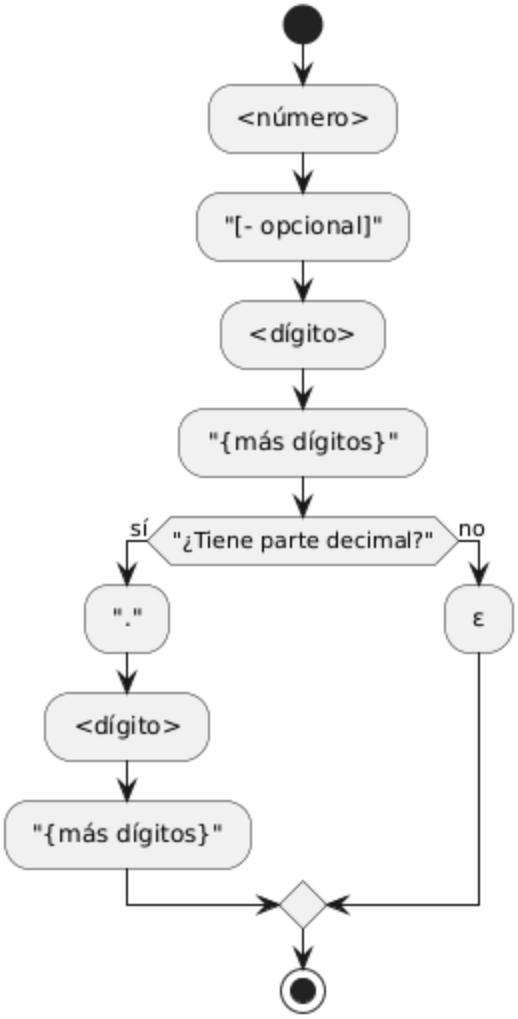

Comentario
<comentario>
"¿Tipo de comentario?"
//
$
%%
/ /
"//"
"$"
"%%"
"/"
<texto>
<texto>
<texto>
<texto>
"%%"
"/"


Valor
<valor>
"Tipo de valor"
cadena
true
false
número
<número>
<cadena>
"true"
"false"

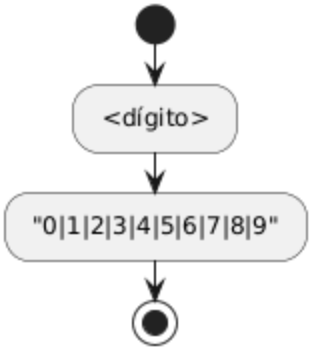

Lista_valores
<lista_valores>
<valor>
<lista_valores'>


Lista_valores'
<lista_valores'>
","
"¿Hay más valores?"
ε
","
ε
<valor>
<lista_valores'>

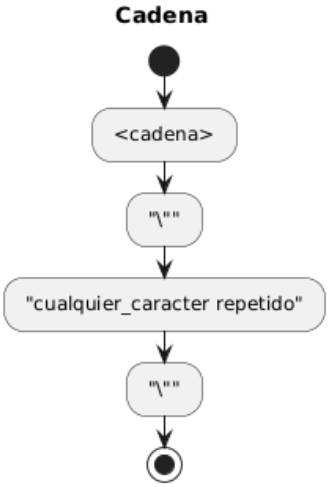
Número



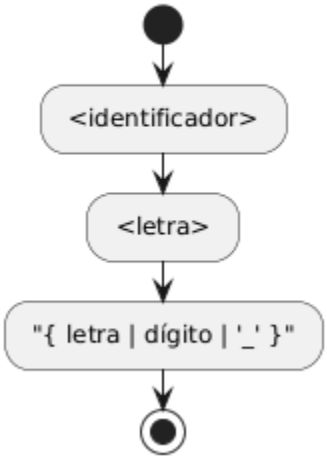
Dígito



Cadena



Identificador



Letra