PROPUESTA INICIAL

Eduardo Bertaud Flores A01138501 Antonio Meza Flores A01175595

1 DE MARZO DE 2016

TECNOLÓGICO DE MONTERREY

Visión del Proyecto

La visión de este proyecto es poder crear una aplicación de output gráfico la cual sea innovadora para todos aquellos que estén interesados en aprender cómo funciona un lenguaje de programación desde una temprana edad.

Objetivo del Lenguaje

El objetivo de este lenguaje es poder enseñarles a jóvenes de secundaria o de preparatoria los fundamentos de la lógica de programación de una manera amigable y entretenida, utilizando elementos como trazos de líneas, curvas, círculos, cuadrados, entre otros más elementos. El área de aplicación a la cual estaría enfocado sería hacia el uso en aulas de educación para enseñar los principios de la programación a jóvenes que les interese el tema de la programación.

Requerimientos del Proyecto

Componentes de léxico del lenguaje

1) PROGRAM
2) END
3) CONST
4) FUNCTION
5) RETURN
6) VAR
7) INT
8) FLOAT
9) BOOL
10) STRING
11) TRUE
12) FALSE
13) PRINT
14) IF
15) ELSE
16) DO
17) WHILE
18) LIST
19) APPEND
20) REMOVE
21) FIRST

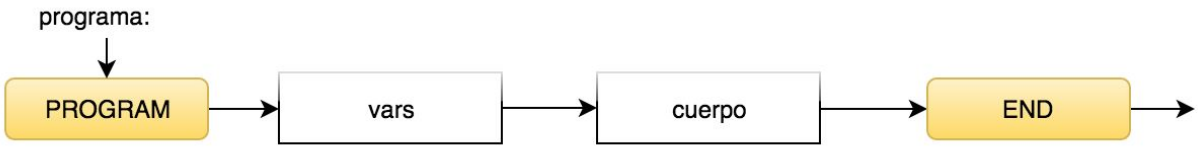
22)LAST
23)SIZE
24)PAINT
25)CREATE
26)ERASE
27)UP
28)LEFT
29)RIGHT
30)DOWN
31)PEN-UP
32)PEN-DOWN
33)CIRCLE
34)SQUARE
35)TRIANGLE
36)LINE
37)POINT

38)COMA	\,
39)PUNTO_COMA	\;
40)DOS_PUNTOS	\:
41)ABRIR_PRNT	\(
42)CERRAR_PRNT	\)
43)ABRIR_LLAVE	\{
44)CERRAR_LLAVE	\}
45)ABRIR_CORCH	\[
46)CERRAR_CORCH	\]
47)SUMA	\+
48)RESTA	\-
49)MULTIPLICACION	*
50)DIVISION	\/
51)IGUAL	\=
52)IGUALDAD	\=\=
53)MENOR_QUE	\<
54)MAYOR_QUE	\>
55)DESIGUALDAD	\<\>
56)ID	[a-zA-Z][a-zA-Z0-9]*
57)CTE_I	[0-9]+
58)CTE_F	[0-9]+\.[0-9]+
59)CTE_S	\".*\"

Diagramas de Sintaxis

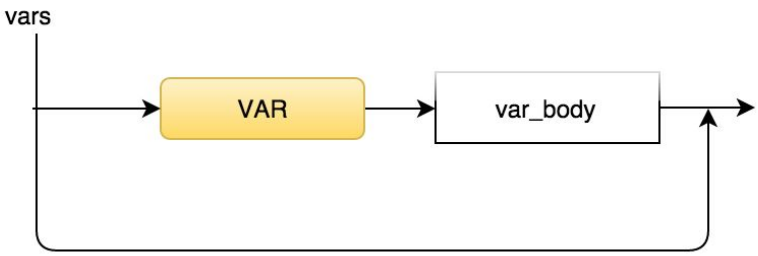
01) program:

PROGRAM vars body END



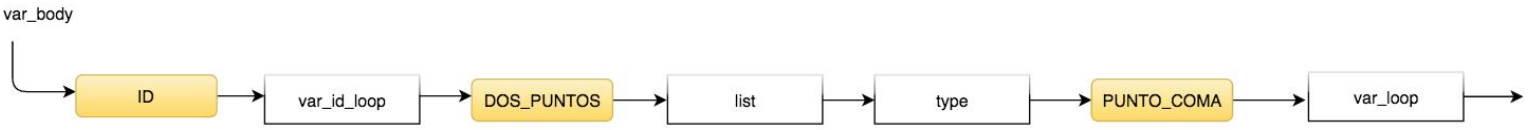
02) vars:

| VAR var_body



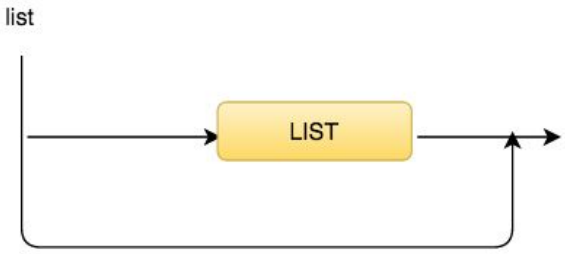
03) var_body:

ID var_id_loop DOS_PUNTOS list type PUNTO_COMA var_loop



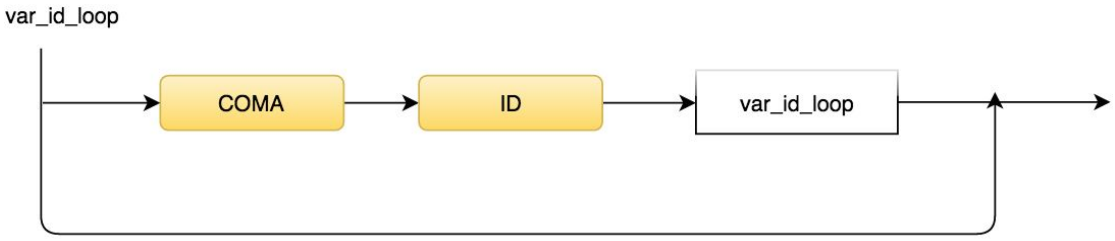
04) list:

| LIST



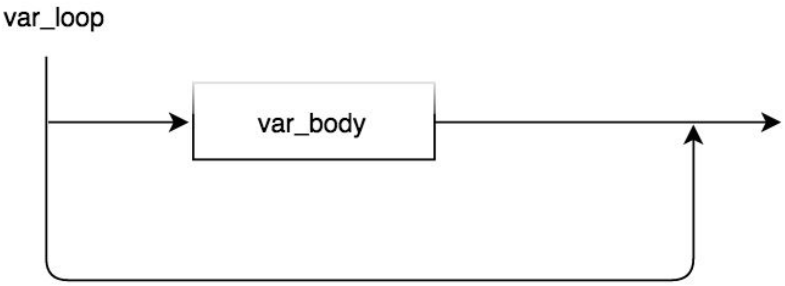
05) var_id_loop:

| COMA ID var_id_loop



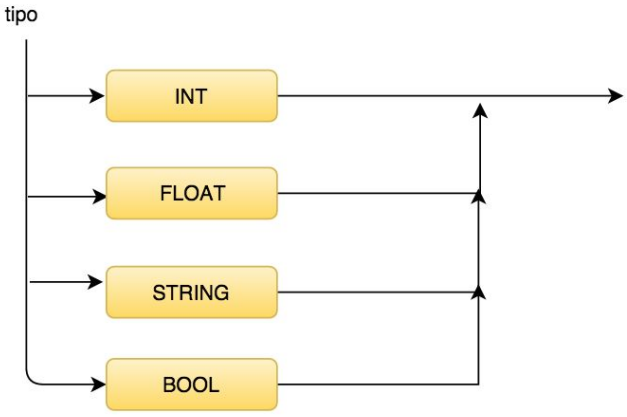
06) var_loop:

| var_body



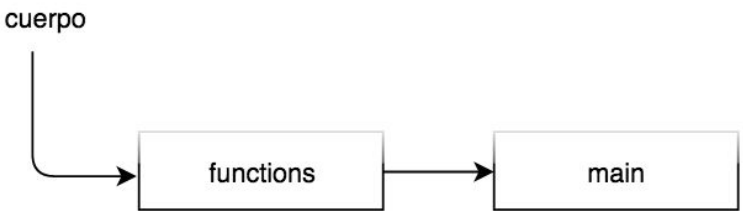
07) type:

INT | FLOAT | STRING | BOOL



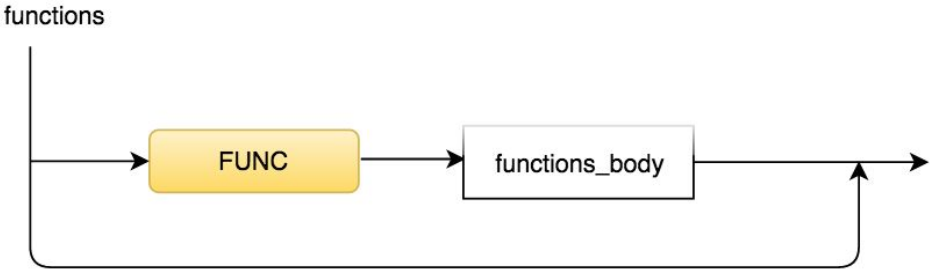
08) body:

functions main



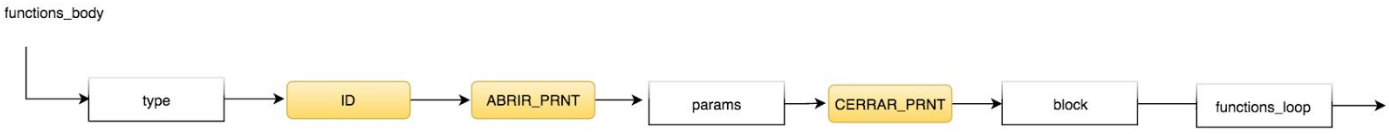
09) functions:

| FUNC functions_body



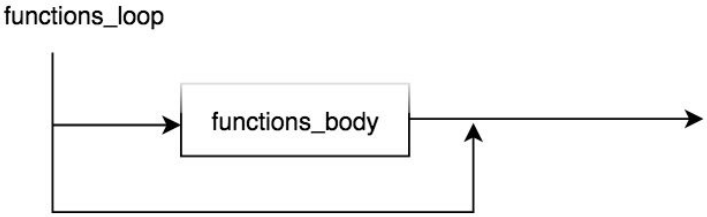
10) functions_body:

type ID ABRIR_PRNT params CERRAR_PRNT block functions_loop



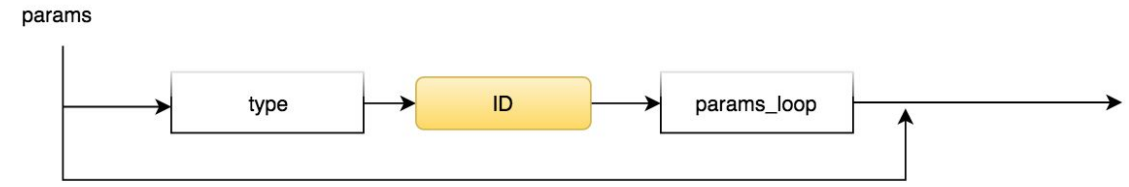
11) functions_loop:

| functions_boody



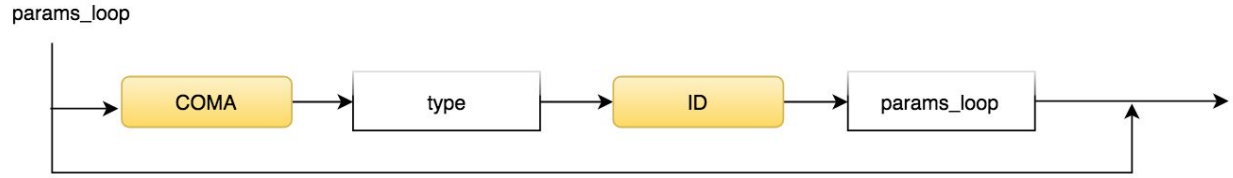
12) params:

| type ID params_loop



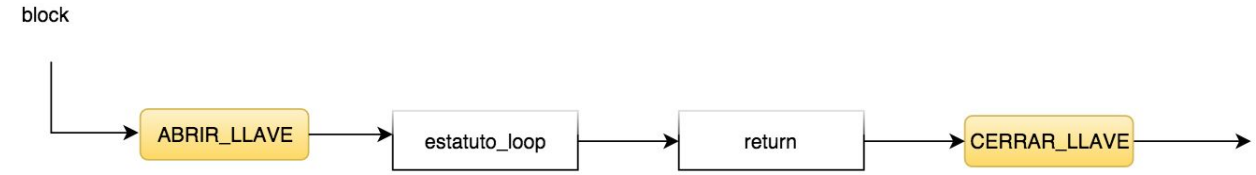
13) params_loop:

| COMA type ID params_loop



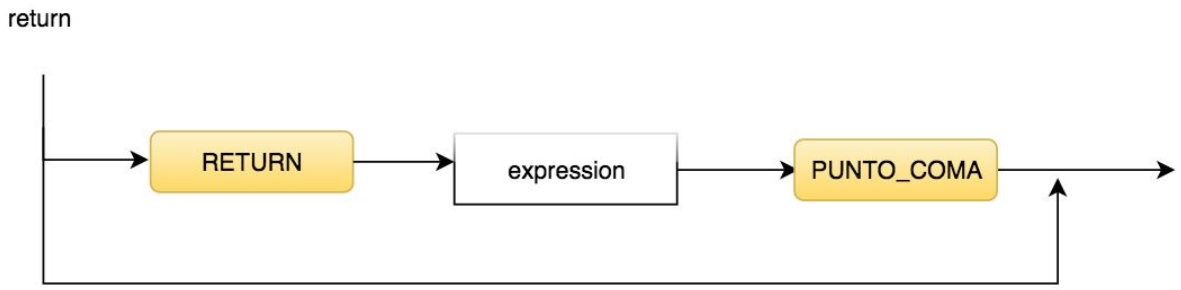
14) block:

ABRIR_LLAVE estatuto_loop return CERRAR_LLAVE



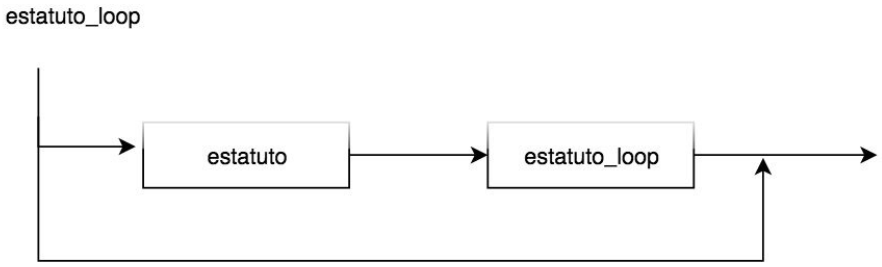
15) return:

| RETURN expression PUNTO_COM



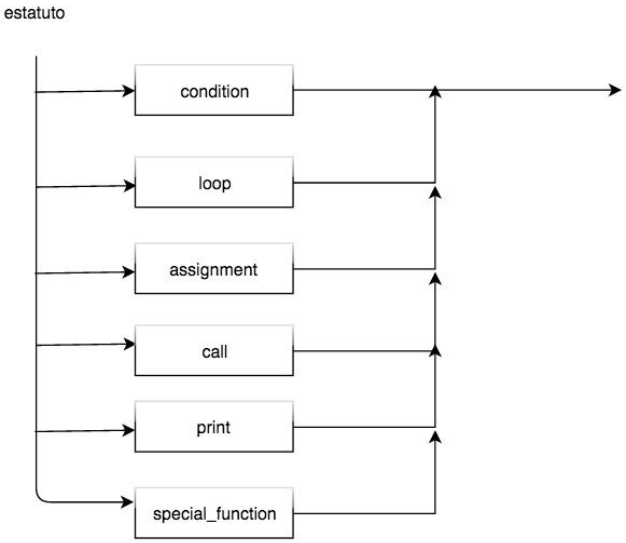
16) estatuto_loop:

| estatuto estatuto_loop



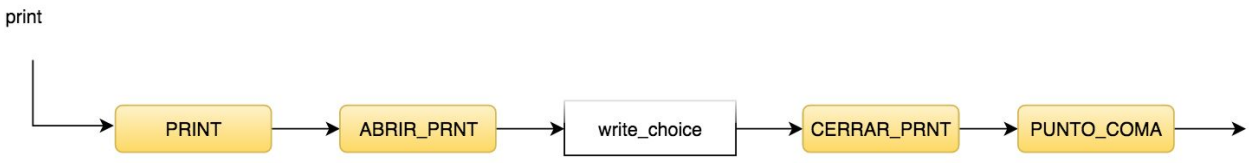
17) estatuto:

condition | loop | assignment | call | print | special_function



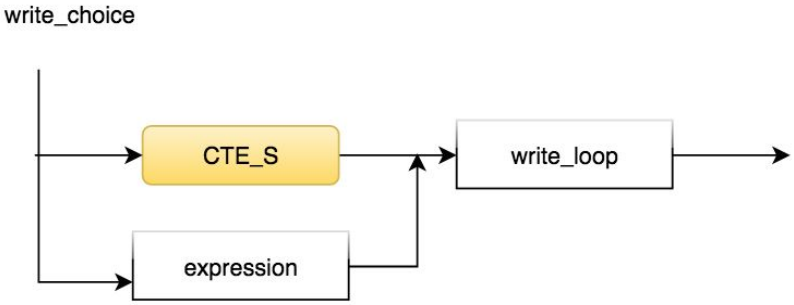
18) print:

PRINT ABRIR_PRNT write_choice CERRAR_PRNT PUNTO_COMA



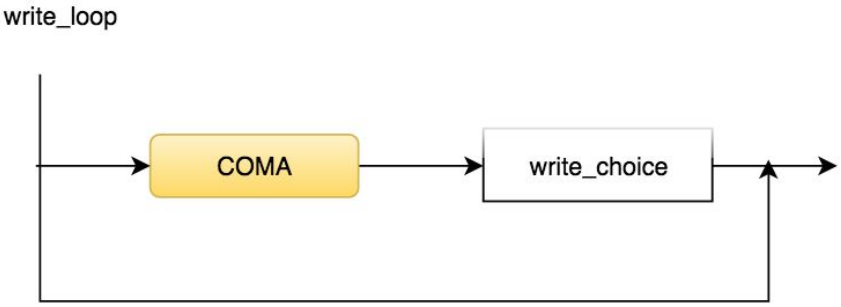
19) write_choice:

expression write_loop | CTE_S write_loop



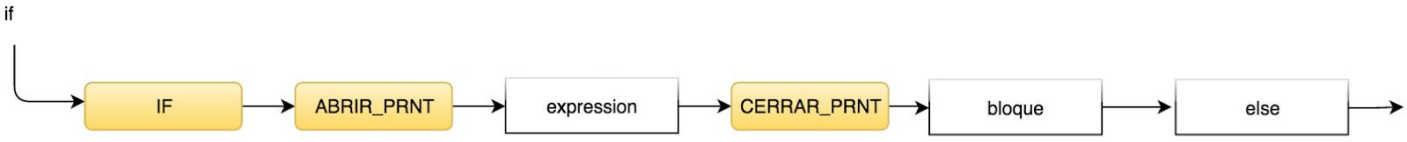
20) write_loop:

| COMA write_choice



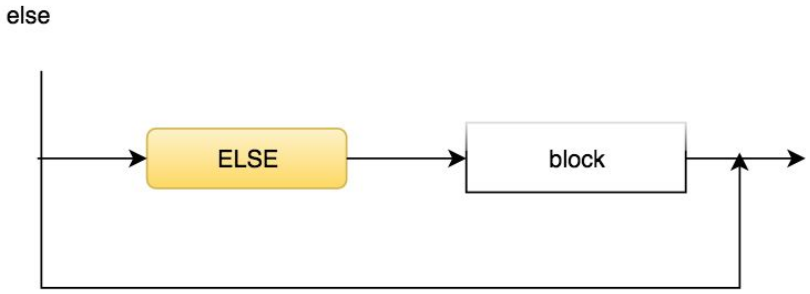
21) condition:

IF ABIR_PRNT expression CERRAR_PRNT block else



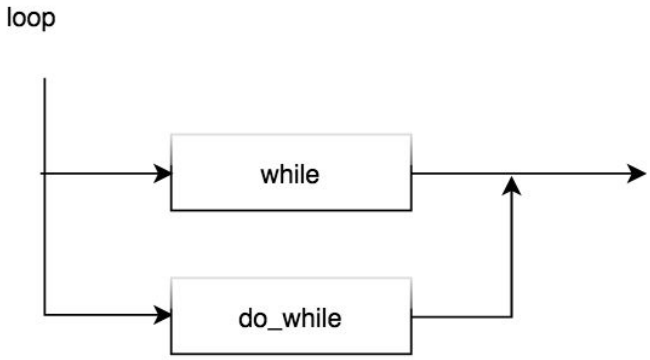
22) else:

| ELSE block



23) loop:

while | do_while

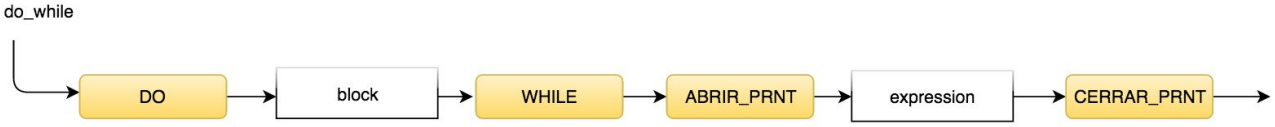


24) while:

WHILE ABRIR_PRNT expression CERRAR_PRNT block

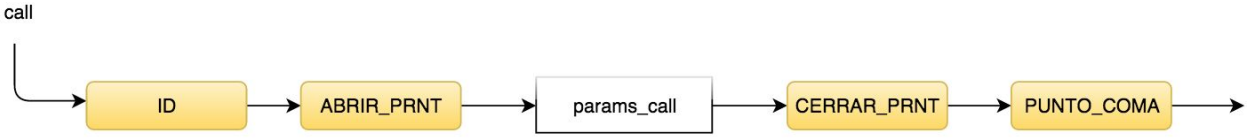
25) do_while:

DO block WHILE ABRIR_PRNT expression CERRAR_PRNT



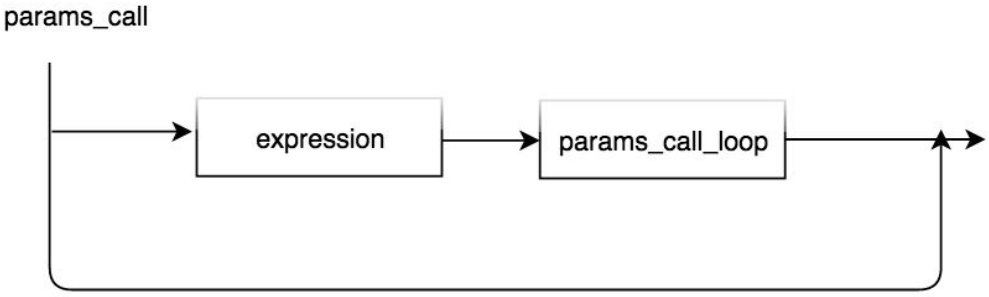
26) call:

ID ABRIR_PRNT params_call CERRAR_PRNT PUNTO_COMA



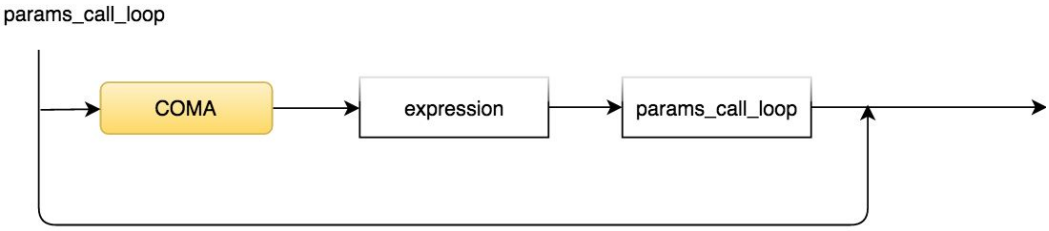
27) params_call:

| expression params_call_loop



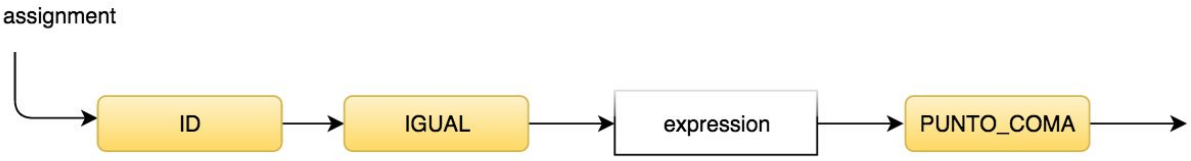
28) params_call_loop:

| COMA expression params_call_loop



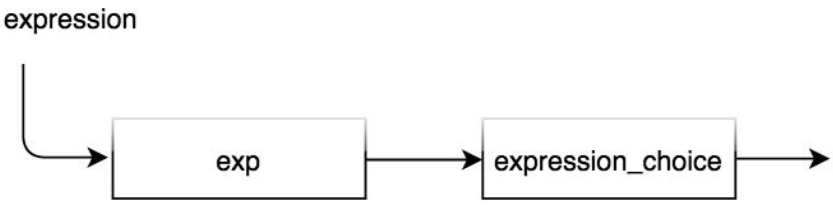
29) assignment:

ID IGUAL expression PUNTO_COMA



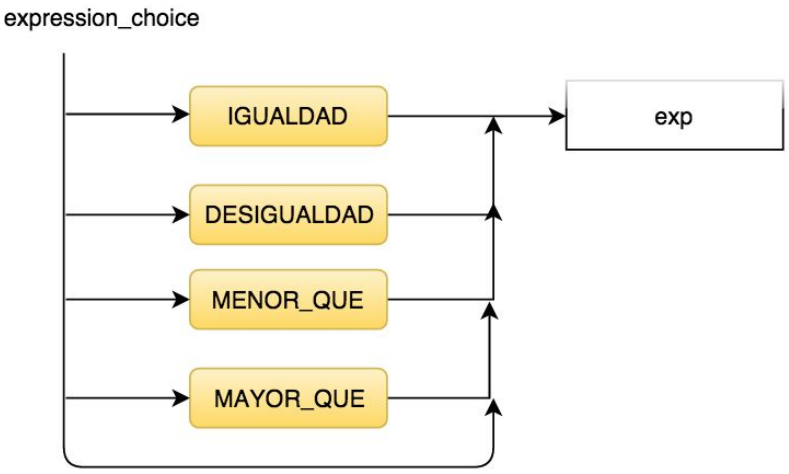
30) expression:

exp expression_choice



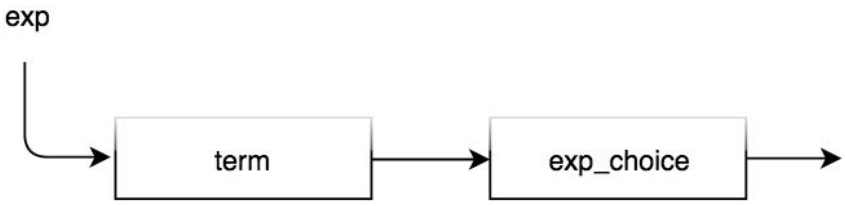
31) expression_choice:

| IGUALDAD exp | DESIGUALDAD exp | MENOR_QUE exp | MAYOR_QUE exp



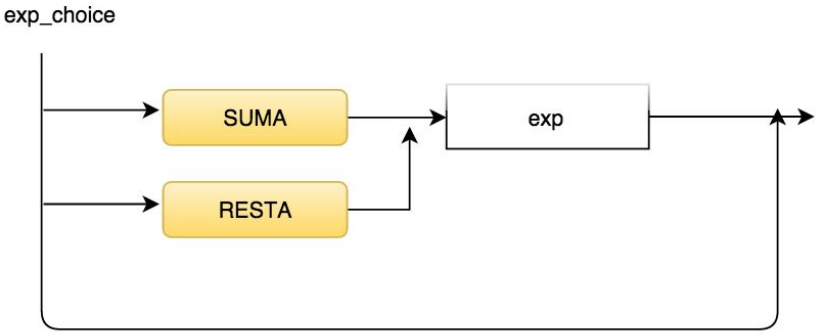
32) exp:

term exp_choice



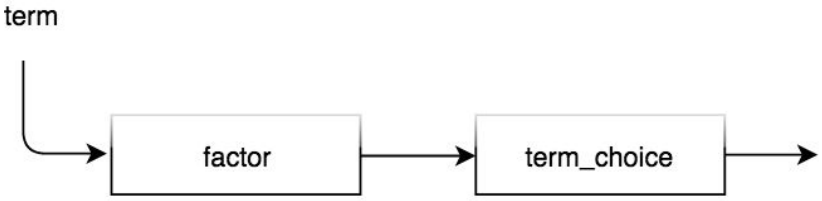
33) exp_choice:

| SUMA exp | RESTA exp



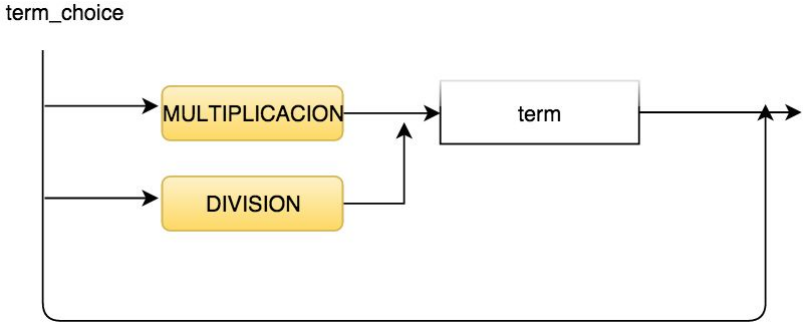
34) term:

factor term_choice



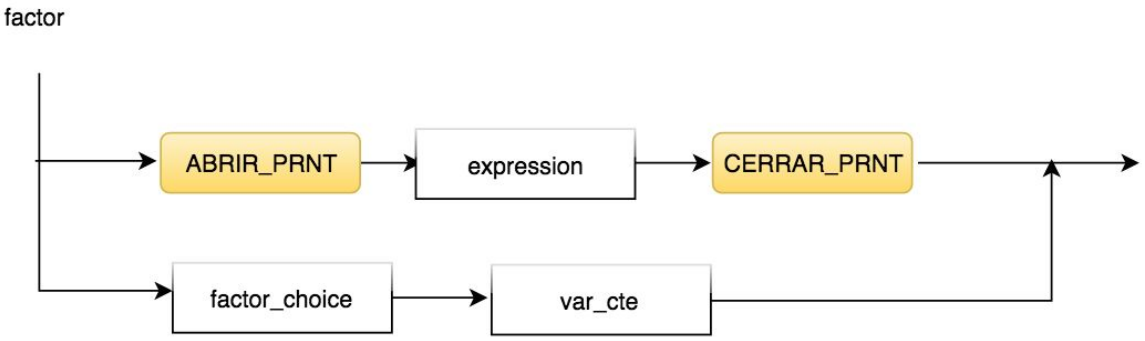
35) term_choice:

| MULTIPLICACION term | DIVISION term



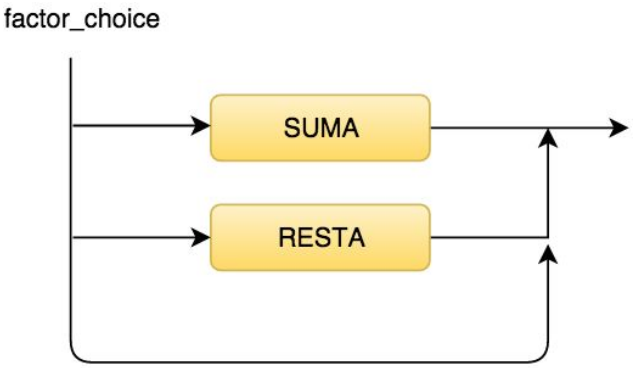
36) factor:

ABRIR_PRNT expresion CERRAR_PRNT | factor_choice var_cte



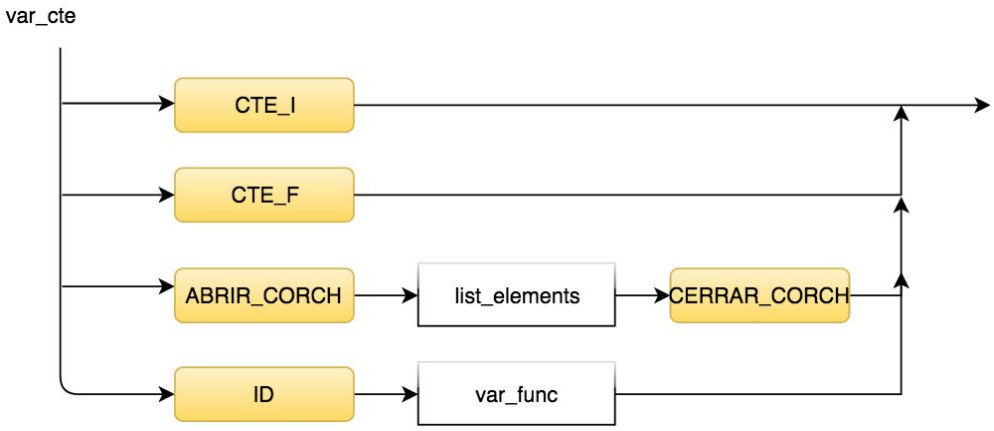
37) factor_choice:

| SUMA | RESTA



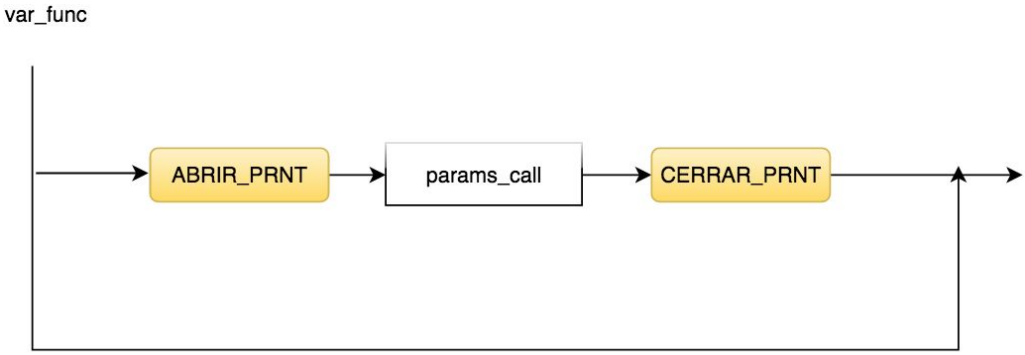
38) var_cte:

CTE_I | CTE_F | ABRIR_CORCH list_elements CERRAR_CORCH | ID var_func



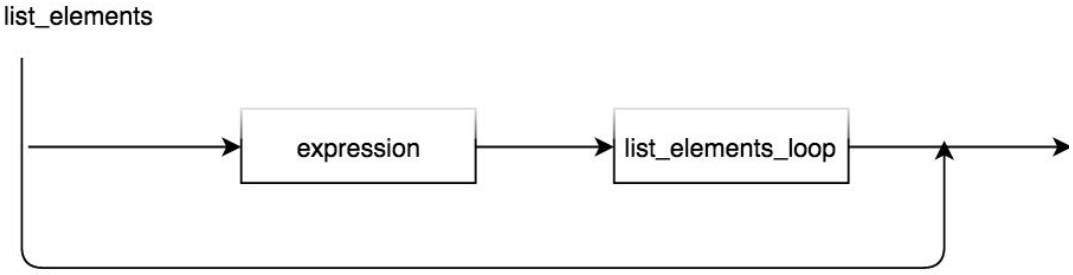
39) var_func:

| ABRIR_PRNT params_call CERRAR_PRNT



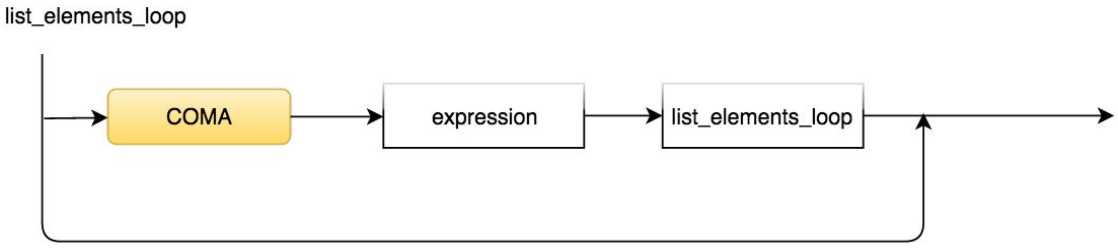
40) list_elements:

| expression list_elements_loop



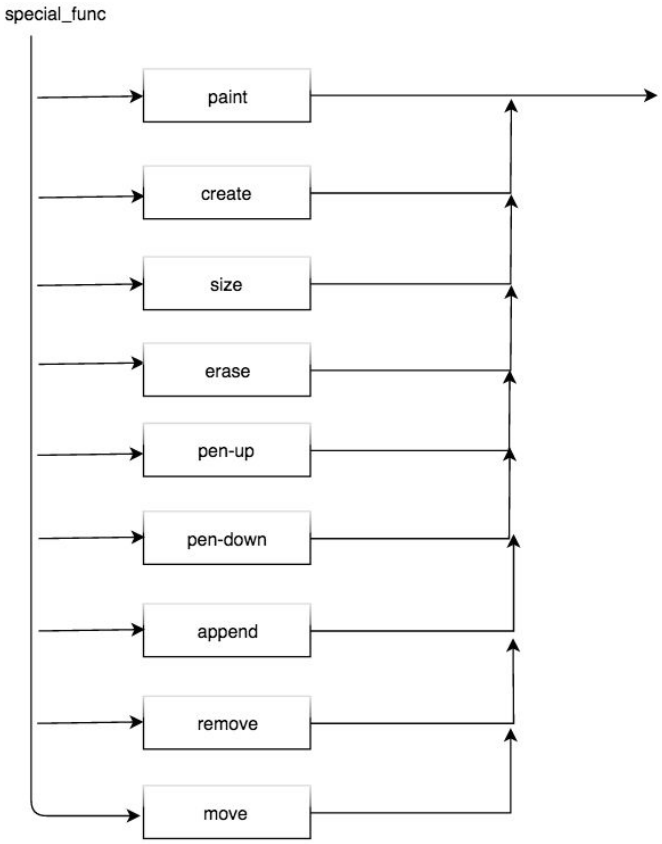
41) list_elements_loop:

| COMA expression list_elements_loop



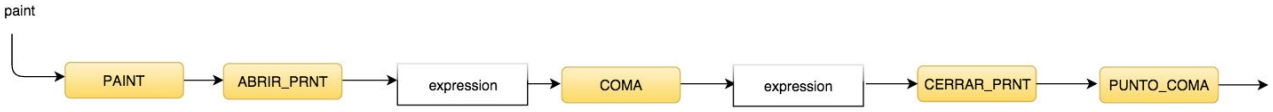
42) special_func:

paint | create | erase | pen-up | pen-down | append | remove | size



43) paint:

PAINT ABRIR_PRNT expression COMA expression CERRAR_PRNT PUNTO_COMA

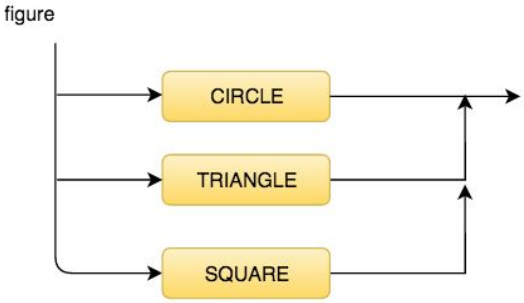


44) create:

CREATE ABRIR_PRNT figura COMA expression COMA expression CERRAR_PRNT PUNTO_COMA

45) figure:

CIRCLE | TRIANGLE | SQUARE

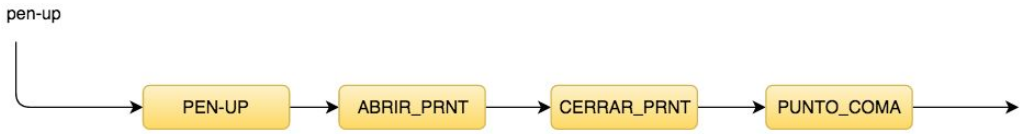


46) erase:

ERASE ABRIR_PRNT expression COMA expression CERRAR_PRNT PUNTO_COMA

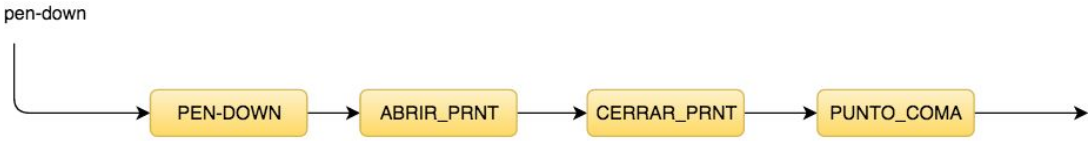
47) pen-up:

PEN-UP ABRIR_PRNT CERRAR_PRNT PUNTO_COMA



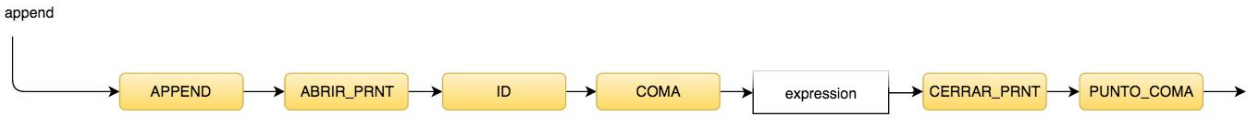
48) pen-down:

PEN-DOWN ABRIR_PRNT CERRAR_PRNT PUNTO_COMA



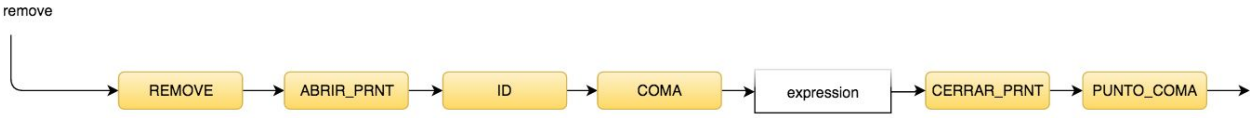
49) append:

APPEND ABRIR_PRNT ID COMA expression CERRAR_PRNT PUNTO_COMA



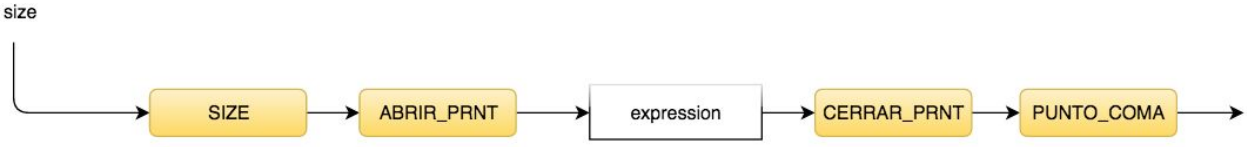
50) remove:

REMOVE ABRIR_PRNT ID COMA expression CERRAR_PRNT PUNTO_COMA



51) size:

SIZE ABRIR_PRNT expression CERRAR_PRNT PUNTO_COMA

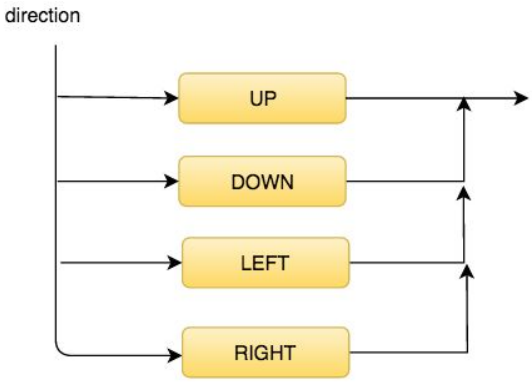


52) move:

MOVE ABRIR_PRNT ditection COMA expression CERRAR_PRNT PUNTO_COMA

53) direction:

UP | DOWN | LEFT | RIGHT



Principales características semánticas

- Un objeto de tipo VAR puede ser una constante entera, una constante flotante o una cadena de caracteres.
- Una variable INT puede ser comparada contra una variable de tipo FLOAT.
- Si se declara una función de tipo VOID esta no regresara ningún valor.
- Si se declara una función de cierto tipo, su valor de retorno deberá de concordar con el tipo de la función.
- La asignación debe de respetar los tipos de variables, no se puede asignar un valor FLOAT a una variable INT.

Descripción de las funciones especiales del lenguaje

Las funciones del lenguaje predefinidas serían las siguientes:

- **PRINT:** Despliega en la terminal mensajes de error.
- **PAINT:** Imprime un punto en pantalla.
- **CREATE:** Crea una nueva instancia de una figura.
- **ERASE:** Remueve un punto de la pantalla.
- **APPEND:** agrega nuevo nodo a la lista.
- **REMOVE:** Remueve un nodo de la lista.
- **SIZE:** Regresa el tamaño de la lista.
- **PEN-DOWN:** Permite dibujar la estela del cursor hacia abajo en la pantalla.
- **PEN-UP:** Permite dibujar la estela del cursor hacia arriba en la pantalla.

Tipos de datos en el lenguaje, incluyendo las limitantes de cada uno de ellos

- **VAR:** puede tener constantes enteras, flotantes, valores booleanos o cadenas de caracteres. Sus limitantes serían que los valores para una variable entera se debe de encontrar entre los rangos -2147483648 a 2147483647, y para variables tipo STRING todas se deben de encontrar entre comillas (“ “). Para las variables BOOL su valor únicamente podrá de ser de 1 y 0.
- **CONST:** puede ser igual a INT, FLOAT, BOOL y STRING.
- **LIST:** Estructura de datos la cual puede almacenar datos de tipo INT, FLOAT, BOOL y STRING.

Plataforma de desarrollo: equipo de cómputo y versión de lenguaje a utilizar

Equipo de computo:

- Dos Macbook Pro retina:
    - Procesador Intel Core i5 dual core de 2.7 GHz
    - 8 GB de memoria integrada LPDDR3 de 1866 MHz
    - Intel Iris Graphics 6100

- OS X El Capitán

Estaremos utilizando el parser PLY y la versión de que se estará utilizando será la 3.8 y el compilador será PYTHON en Mac OS X donde la versión que se utilizara será la 3.5.1.

Bibliografía

- Logo Interpreter. (n.d.). Retrieved February 26, 2016, from http://www.calormen.com/jslogo/
- PLY (Python Lex-Yacc). (n.d.). Retrieved February 26, 2016, from http://www.dabeaz.com/ply/