TECNOLÓGICO DE MONTERREY CAMPUS MONTERREY

TECNOLÓGICO
DE MONTERREY®

# Propuesta: ViLe (Visual Learning)
# Diseño de compiladores

29/02/2016

Marcel Benítez Martínez A01139855
Oscar Abraham Rodriguez Quintanilla A01195653

## Visión del proyecto

Nuestra visión es que cualquier persona que quiera aprender a programar desde cualquier parte del mundo pueda hacerlo de manera fácil y accesible.

## Proposito

El propósito de este lenguaje es que las personas que estén interesadas en aprender a programar puedan hacerlo de manera fácil y visual. La característica principal es que estará en la web, haciendo que se pueda acceder de manera rápida con tan solo teniendo acceso a internet.

## Objetivo del lenguaje

El objetivo de nuestro lenguaje de programación, es que las personas, en especial los jóvenes, aprendan las bases de la programación de una manera sencilla y visual, para que después de que vean lo fácil que es transformar lo que se piensa hacer a código y acciones de la computadora, se interesen más en el mundo de la programación y/o entiendan cómo es que las cosas que usan diariamente funcionan.  El área de aplicación de nuestro lenguaje es meramente de aprendizaje y educación para cualquier persona interesada en el mundo de la programación.

## Requerimientos del lenguaje

***Componentes del léxico del lenguaje***

Palabras reservadas:

```
def : FUNCTION
main : MAIN
return : RETURN
void : V
if : IF
else : ELSE
print: PRINT
int: INT
float: FLOAT
```

```
string: STRING
list: LIST
bool: BOOL
true: TRUE
false: FALSE
WHILE: while
TIMES : times
AND: and
OR: or
```

Tokens:

```
+  : SUM
-  : MINUS
*  : MULTIPLY
/  : DIVIDE
%  : MOD
=  : EQUALS
== : EQUALITY
<  : GREATER
>  : LESS
<= : LESS_EQUAL
>= : GREATER_EQUAL
!= : DIFFERENT
;  : SEMICOLON
,  : COMA
(  : O_PARENTHESIS
)  : C_PARENTHESIS
{  : O_BRACKET
}  : C_BRACKET
[  : O_S_BRACKET
]  : C_S_BRACKET
```

Expresiones regulares

".*" : STRINGCONST

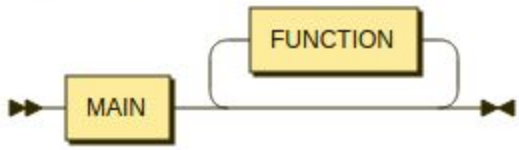
-?[0-9]+ : INTCONST

-?[0-9]+.+[0-9]+ : FLOATCONST

[a-zA-Z]+(_?[a-zA-Z0-9])* : ID

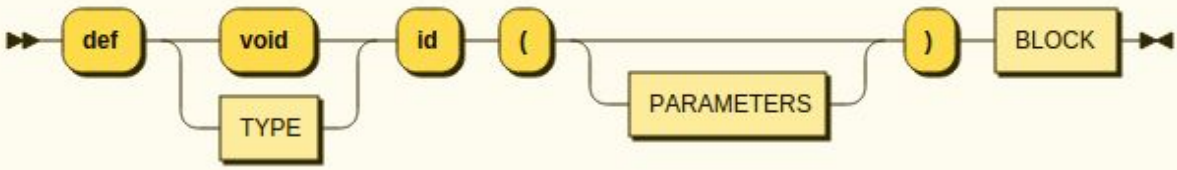
***Diagramas de Sintaxis***

**PROGRAM:**

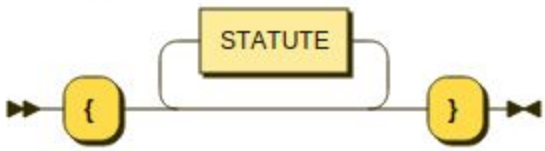



**MAIN:**

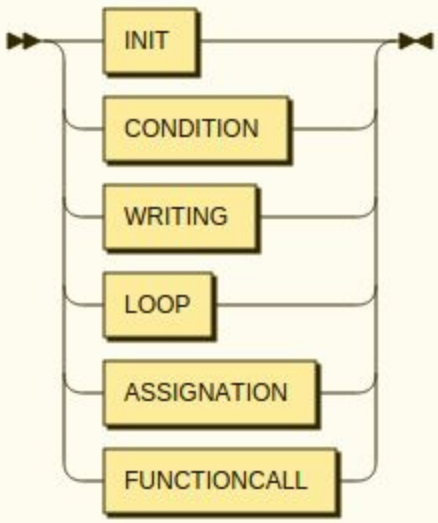

**FUNCTION:**




**BLOCK:**

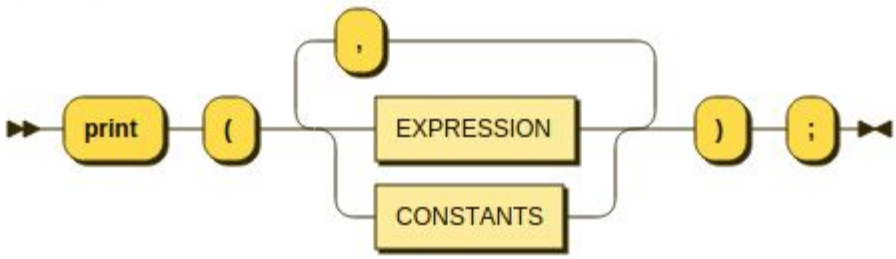


**STATUTE:**

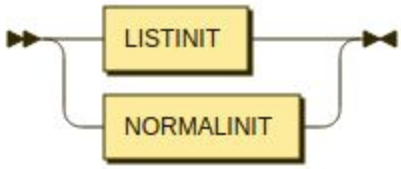



**ASSIGNATION:**

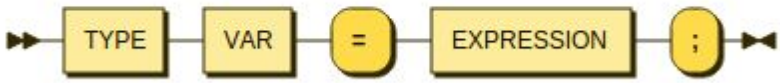

**WRITING:**




**INIT:**

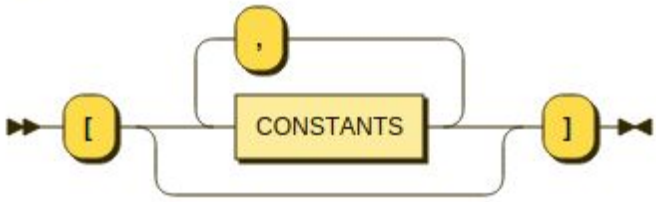


**NORMALINIT:**

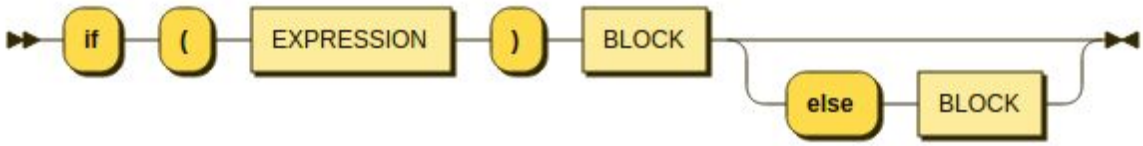



**LISTINIT:**

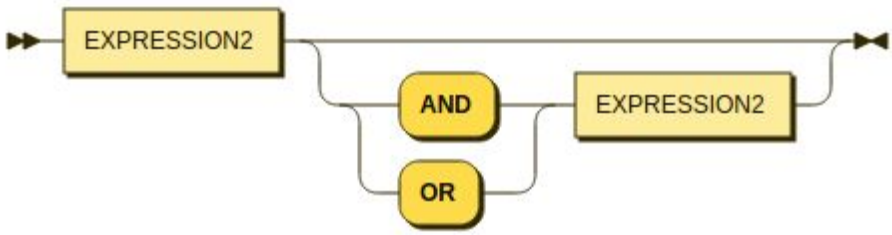

**LIST:**



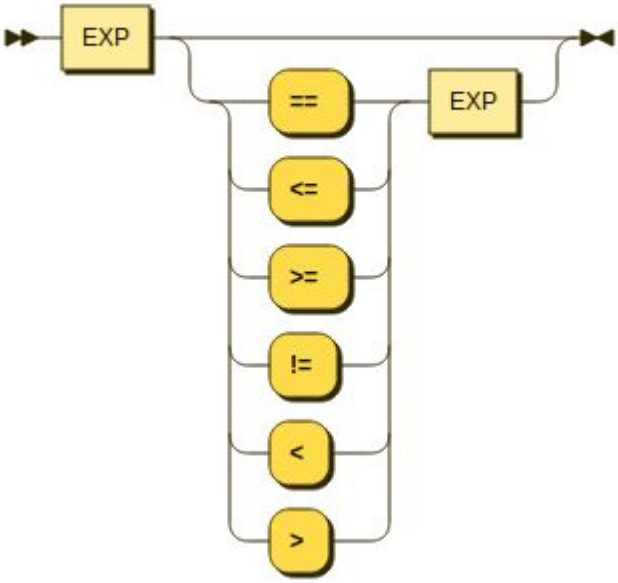

**CONDITION:**



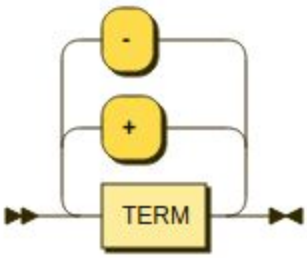

**EXPRESSION:**



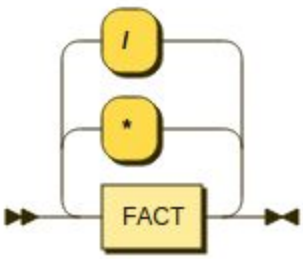
**EXPRESSION2:**



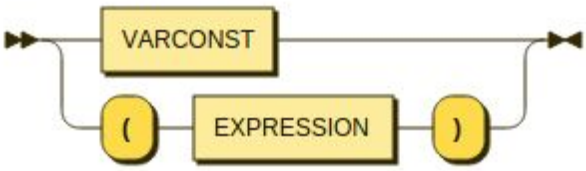

**EXP:**



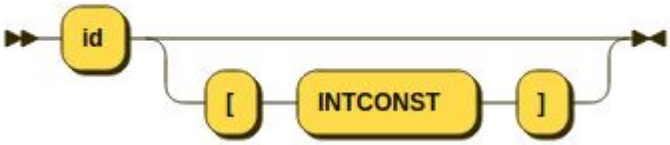

**TERM:**



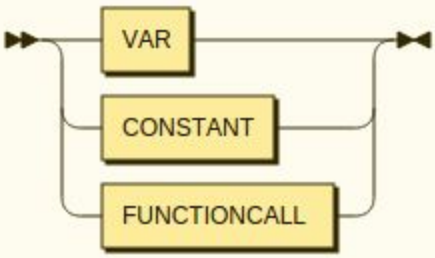
**FACT:**



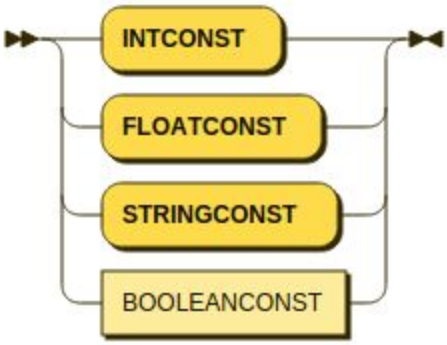

**VAR:**



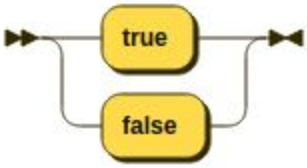

**VARCONST:**



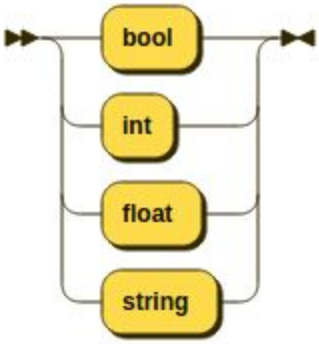

**CONSTANTS:**



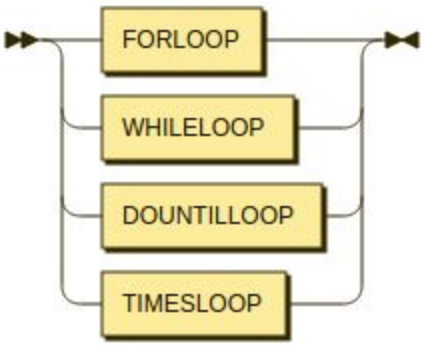
**BOOLEANCONST:**




**TYPE:**

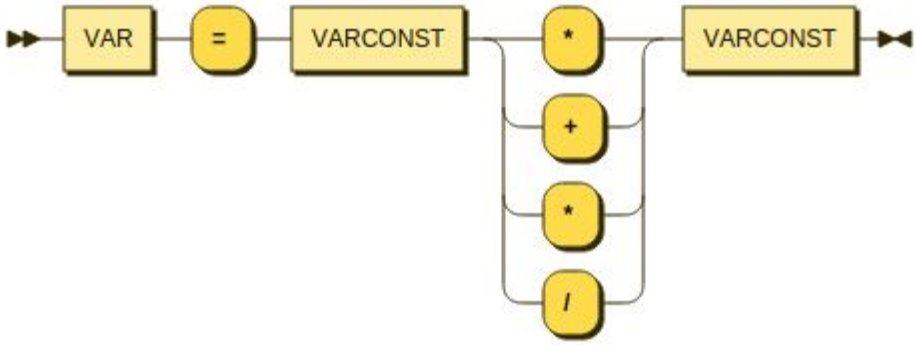



**LOOP:**



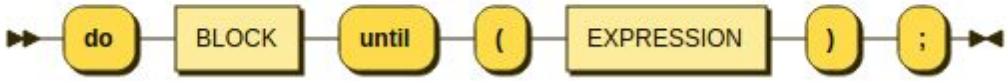

**FORLOOP:**

**INCREMENTORDECREMENT:**



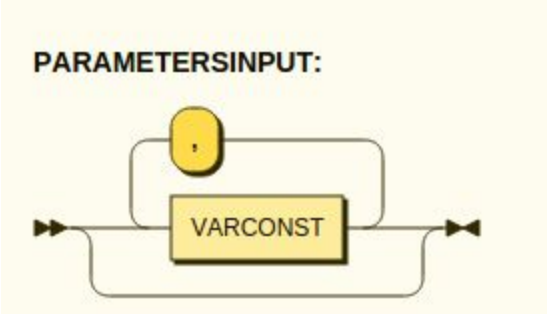

**WHILELOOP:**

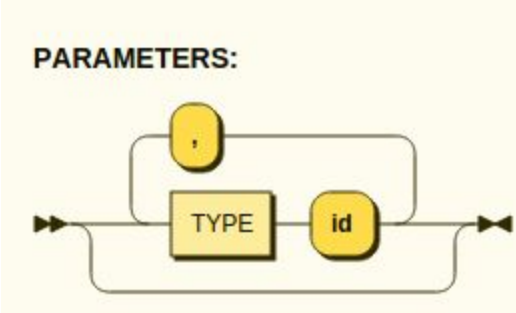

**DOUNTILLOOP:**




**TIMESLOOP:**


**FUNCTIONCALL:**




***Principales características semánticas***

Para las tablas de variables usaremos dos diccionarios de python. Uno de estos tendrá toda la información global del programa, es decir, guardará las direcciones de las funciones y las variables globales, mientras que el segundo diccionario será temporal y solo existirá mientras se esté en una función.

Decidimos usar esta estructura de datos porque se hace más eficiente la búsqueda de variables cada vez que se encuentra una, ya que con esta estructura de datos podemos usar un get('llave') y si no existe simplemente regresa un mensaje de error.

***Descripcion de las funciones especiales del lenguaje***

Para facilitar el aprendizaje de los usuarios, implementaremos una función nueva que se llamará "`times`". Times será un ciclo similar al while, pero más comprehensible para la lectura. Funcionará de la siguiente manera:

```
times(n) {
      //código
}
```

Lo que hará es iterar n veces el código que se encuentre entre los brackets. Es como un for de 1 a n, o un while con un contador hasta n.

El resto de las funciones funcionarán de la misma forma que en lenguajes comunes.

***Tipos de Datos en el lenguaje***

**Enteros:** se definirán en el bloque de variables con la palabra reservada int. Sólo aceptará números enteros positivos y negativos. También se podrán usar como constantes.

**Flotantes:** se definirán en el bloque de variables con la palabra reservada float. Aceptará números negativos y positivos con precisión de 53 bits, al igual que python. También se podrán usar como constantes.

**Strings:** se definirán en el bloque de variables con la palabra reservada string y estas deben estar entre comillas. `Ejemplo: "un string".` Aceptará cadenas de caracteres , al igual que python. También se podrán usar como constantes.

**Booleanos:** se definirán en el bloque de variables con la palabra reservada bool y estas deben de tener como valor True o False. `Ejemplo: bool b = True;`. También se podrán usar como constantes usando True o False.

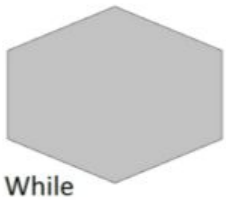
**Listas:** se definirán en el bloque de variables con la palabra reservada list, seguidas por el tipo de dato que manejará esta lista. Ejemplo: `list int lista [1, 2, 3, 4];print(lista[1])`. Esta variable será estática, una vez declarada y teniendo asignados los valores, no se podrán agregar más, solo modificar. Sólo podrá ser usada para accesos o modificaciones a su información en el índice deseado. Ejemplos: `print(lista[1]) ; lista[1] = 3 + 2;`

## Componentes de la parte visual

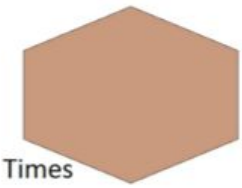
Variables

Asignación

If



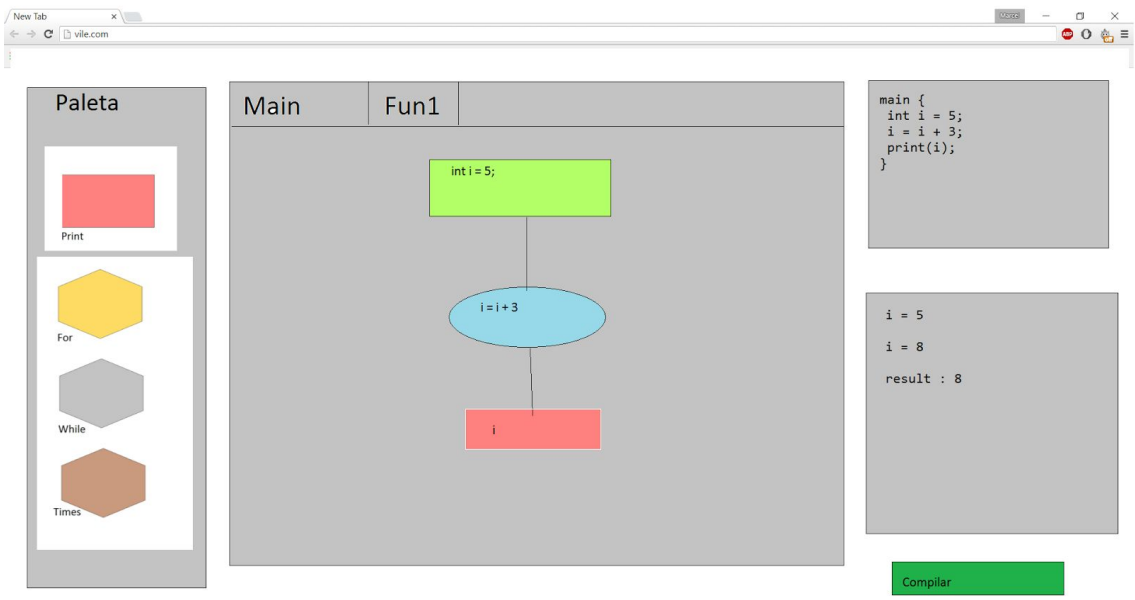
While



Times

Print

## Ejemplo:



## Plataforma de desarrollo

Como nuestro proyecto es basado en web, haremos uso de Ruby on Rails para el backend de nuestra aplicación, esto porque debemos de enlazar nuestra aplicación visual con nuestro compilador que será desarrollado en python, haciendo uso de PLY como analizador sintáctico y léxico, para una más fácil elaboración de este mismo. La parte visual de nuestro proyecto será javascript, html y css.

La aplicación permitirá a los usuarios realizar diagramas de flujo estilo raptor en sus exploradores. Una vez que terminan y presionan el botón compilar, se manda el html con el diagrama al servidor de rails, el cual traducirá este html a nuestro lenguaje, para después ser compilado.

***Componentes y versiones:***

| Componente | Version |
|---|---|
| Rails | 4.2.4 |
| Ruby | 2.2.1 |
| Python | 2.7 |
| Javascript - Coffeescript | 1.7, 4.1.0 |
| HTML | 5 |

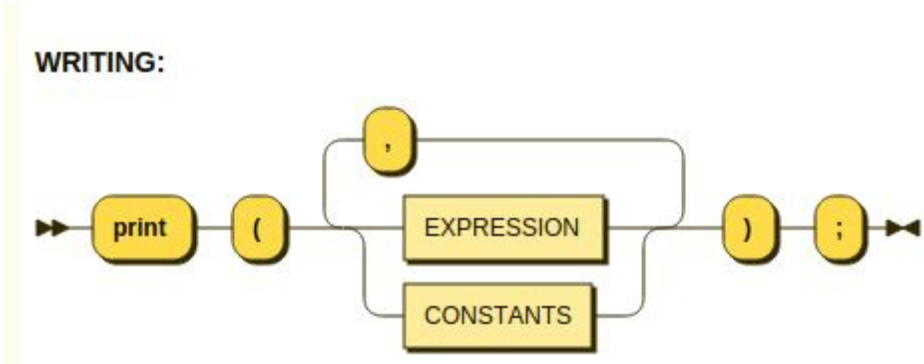
CORRECCIONES

Remover for, do, until, var y colon del léxico del lenguaje y quitarlos de componentes

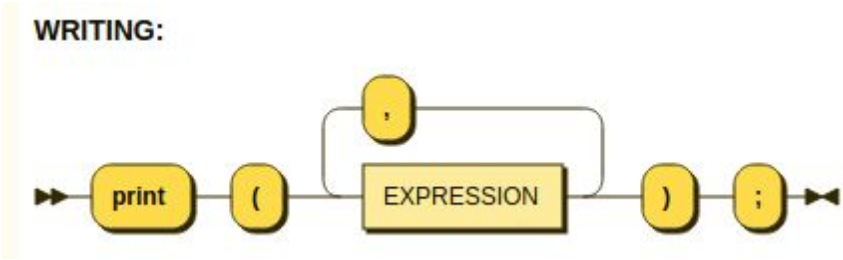
visuales.

Quitar colon y var del léxico del lenguaje.

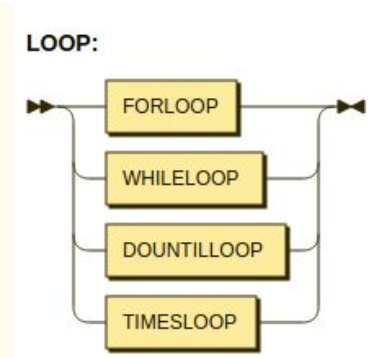
Anterior:



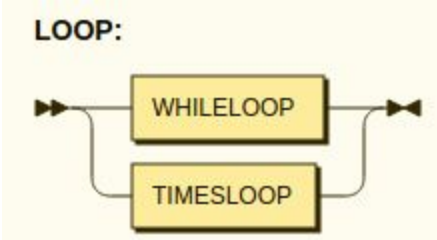

Nuevo:



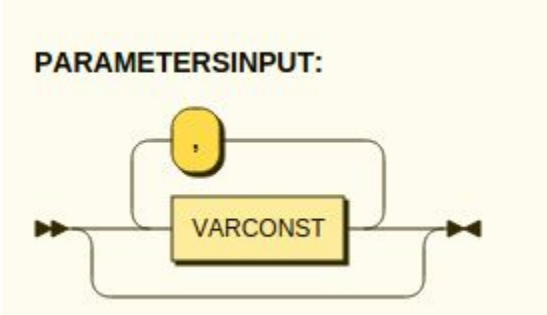

Anterior:



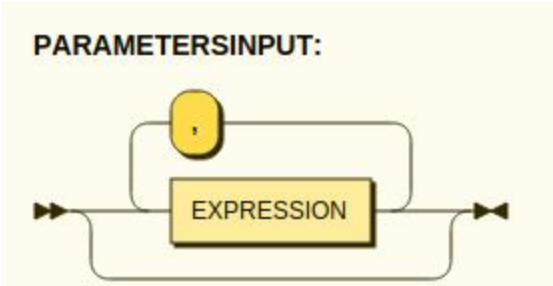

Nuevo:



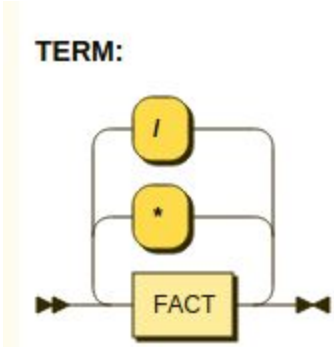

Anterior:

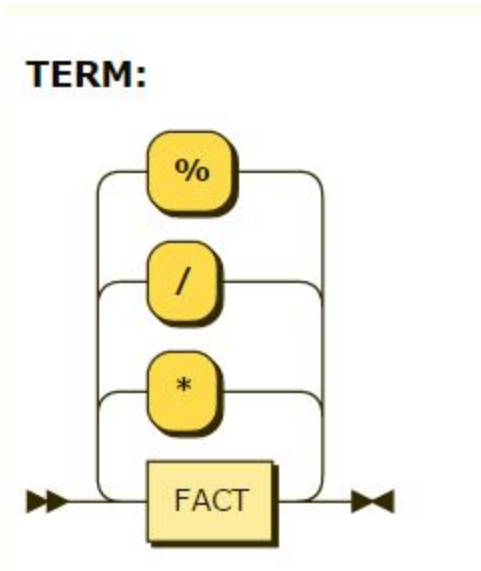



Nuevo:

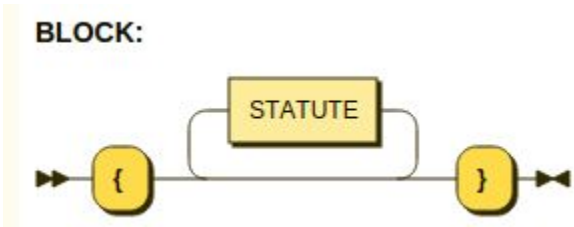



Antes:






Antes:




Nuevo: