Caso de estudio

Operaciones aritméticas

+	+
-	-
*	*
/	/
MOD	%

Operaciones Relacionales

>	>
<	<
Comparación	==
Asignación	=
<=	<=
>=	>=
hasta	~

Operadores lógicos

AND	&&
OR	| |

Condicionales

Condición simple	Operando Op.realcional operando
Condición compuesta	CondiciónSimple Op.Logico CondiciónSimple
if	if(condicion) *contenido if* *eif*
If else	if(condicion) *contenido de if* else_ *contenido else* eelse_ eif_

	}
Condición simple For	Operando ~ operando

Tipos de datos

Entero	int
String	string
float	float

Ciclos

for	for(**CondiSimple; CondicionSimpleFor; condiSimple)** **contenido del for** efor
while	while_(CondicionSimple/compuesta) contenido while ewhile_

Alfabeto

Letras minusculas

Letras de la a-z

Números 0-9

(

)

Op lógicos

Op relacionales

Op aritméticos

“

”

;

~

PALARBAS RESERVADAS

int

string

float

if

eif

else

eelse

for

while

eewhile

escribir

leer

inicio

fin

GRAMATICA MI LENGUAJE

G = { TERMINALES }, { NO TERMINALES }, SÍmbolo de inicio, { REGLAS DE PRODUCCION }

G = {<programa>, {0,1,2,3,…..9, a, b, c….., z,~, +, -, /, *, % , (,),&, |, !, escribir, if, eif,

else, eelse, for, efor, while, ewhile, leer, int, string, float, inicio, fin}

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

<letra>:: a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

<ID>:: <letra> | <ID> <letra>_

<OpAritmetico>:: +|-|*|/|%|=

<OpLogico>:: &&| ||

<OpRelacional>:: <|>|!|!=|==|<=|>=

<tipo>:: int|string|float

<CteInt>:: <nums> | <CteInt> <nums>

<CteFloat>:: <CteInt>.<CteInt>

<CteString>:: “/.”

<decla>:: <tipo> <ID> ;

<declara>:: <decla>|<decla><declara>

<constante>:: <CteInt>|<CteFloat> | <CteString>

<operando>:: <ID>|<constante>

<EXP>:: <operando>| <operando>+<operando>

<Asig>:: <ID> = <operando>

<OperacionArit>:: <ID> = <operando> <OpAritmetico> <operando>

<condicionSimple>:: <operando> <OpRelacional> <operando>

<condicionSimpleFor>:: <operando> ~ <operando>

<condicion>::<condicionSimple>| <condicion><OpLogico><condicionSimple>

<leer>:: leer <ID>;

<escribir>:: escribir <EXP>

<ends>:: ewhile|eif|eelse|efor

<if>:: if (<condicion>)

<else>:: else

<for>:: for (<Asig>; <condicionSimpleFor>; <OperacionArit>)

<while>:: while (<condicion>)

<forr>:: for (“.*”)

<whilee>:: while (“.*”)

<iff>:: if (“.*”)

<ini>:: inicio

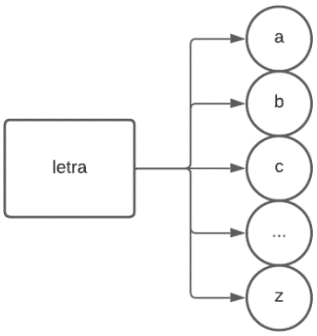
<fin>:: fin

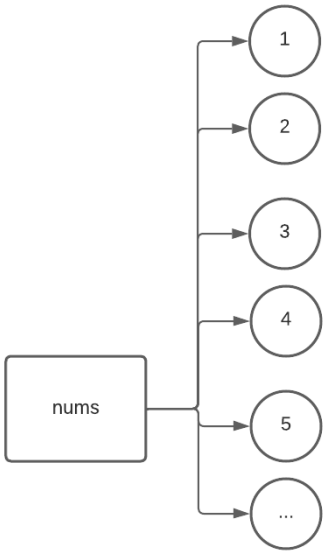
<vacio>::

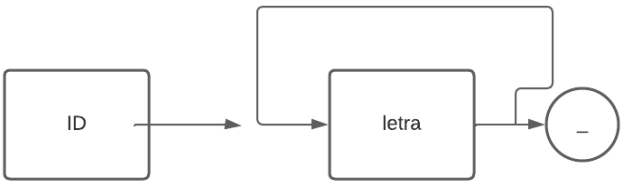
<sentencia>::
<leer>|<escribir>|<if>|<else>|<for>|<while>|<ends>|<Asig>|<declara>|<OperaciónArit>|<fin>|<inicio>|<vacio>

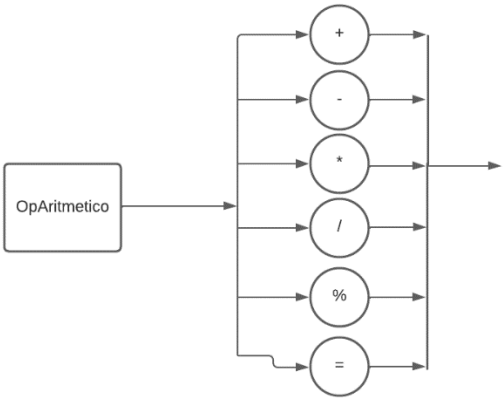
<s>:: <sentencia> | <s><sentencia>

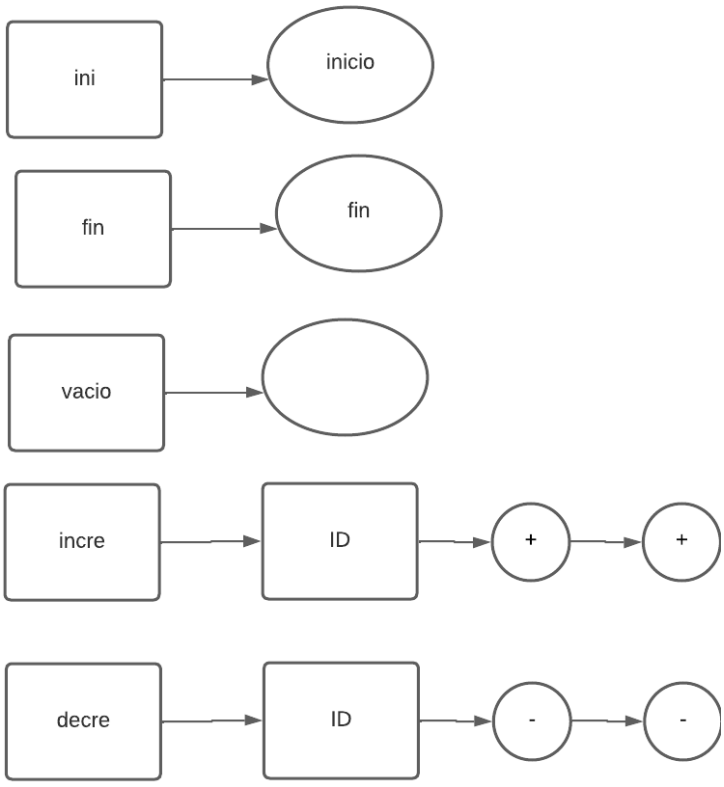
<programa>:: <s>

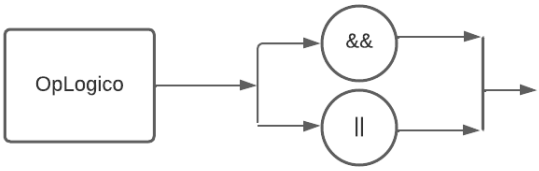

letra
a
b
c
...
z

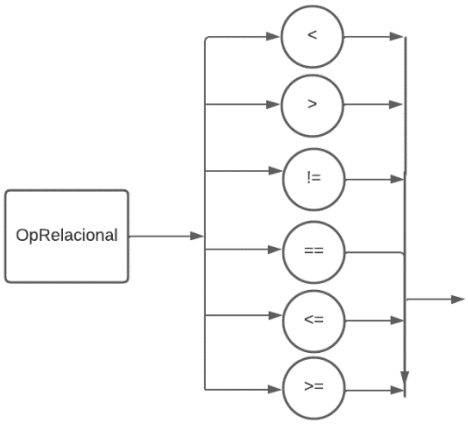

nums
1
2
3
4
5
...

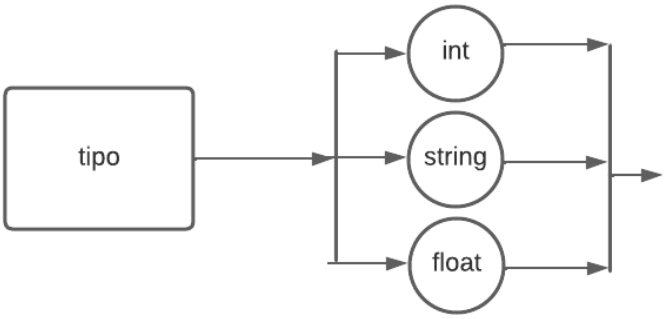

ID
letra
_

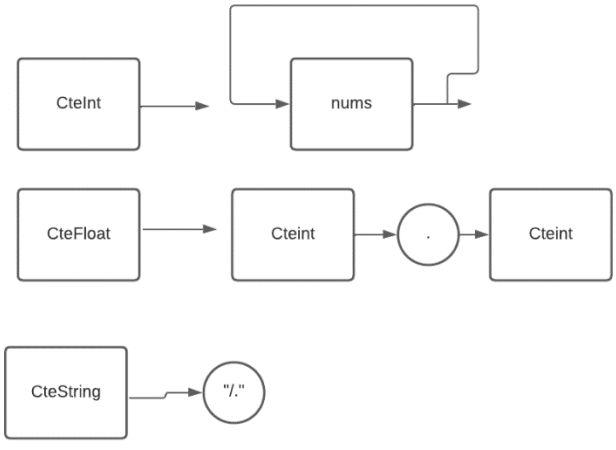

OpAritmetico
+
-
*
/
%
=

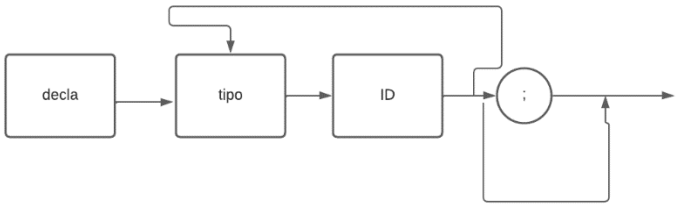

ini
inicio
fin
fin
vacio
incre
ID
+
+
decre
ID
-
-

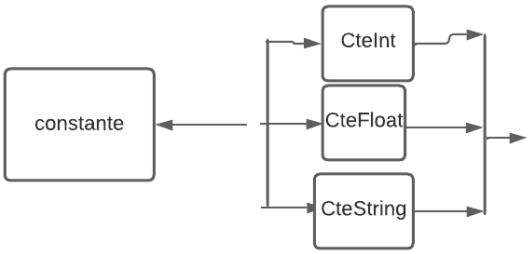

OpLogico
&&
||

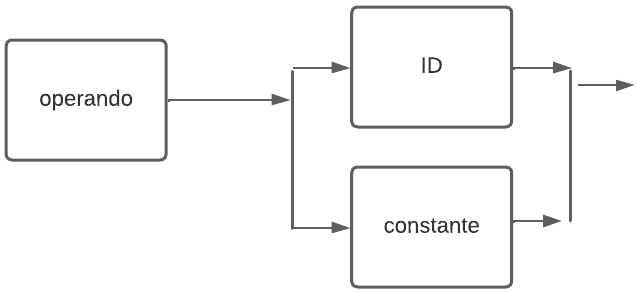

OpRelacional
<
>
!=
==
<=
>=

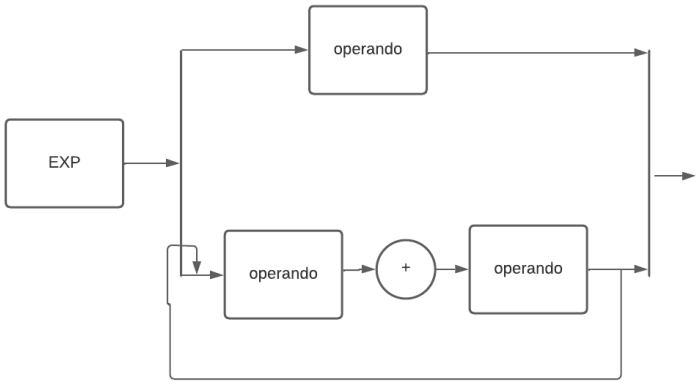

tipo
int
string
float

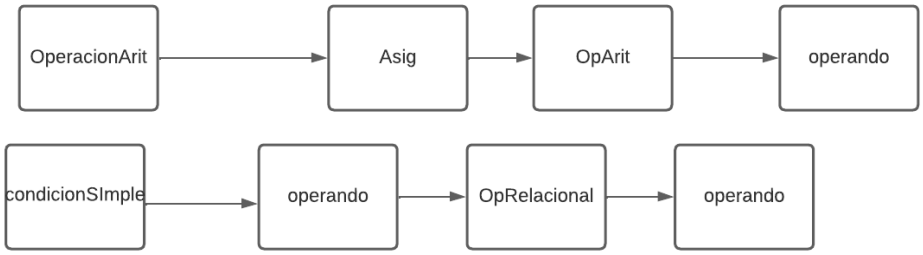

CteInt
nums
CteFloat
Cteint
.
Cteint
CteString
"/."

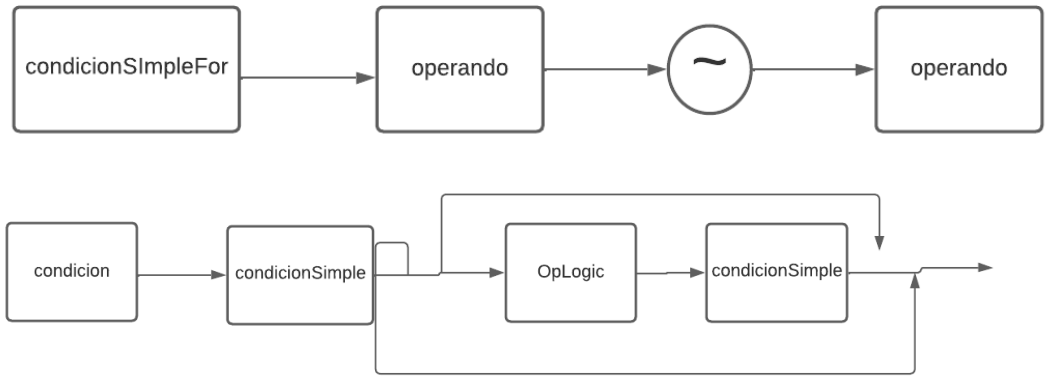

decla
tipo
ID
;

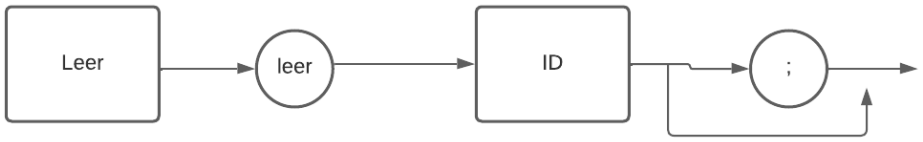

constante
CteInt
CteFloat
CteString

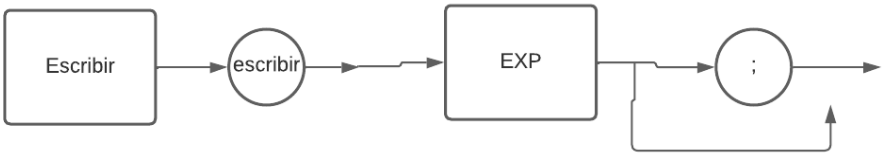

operando
ID
constante

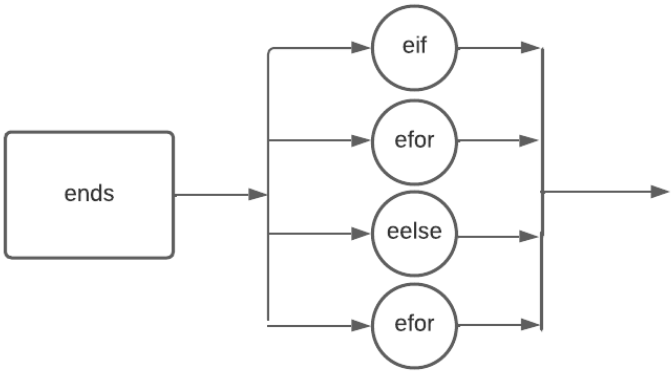

operando
EXP
operando
+
operando

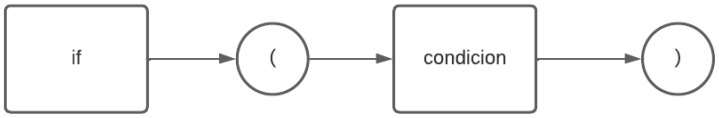

OperacionArit
Asig
OpArit
operando
condicionSImple
operando
OpRelacional
operando

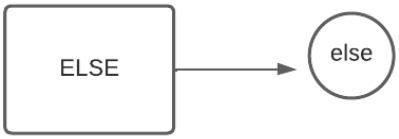

condicionSImpleFor
operando
~
operando
condicion
condicionSimple
OpLogic
condicionSimple

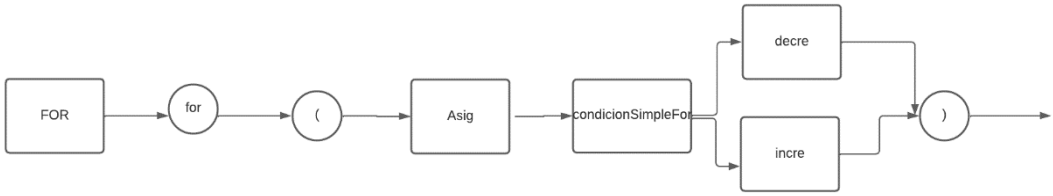

Leer
leer
ID
;

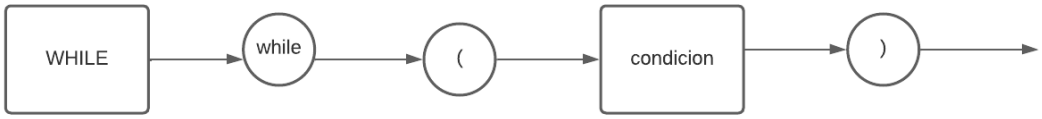

Escribir
escribir
EXP
;

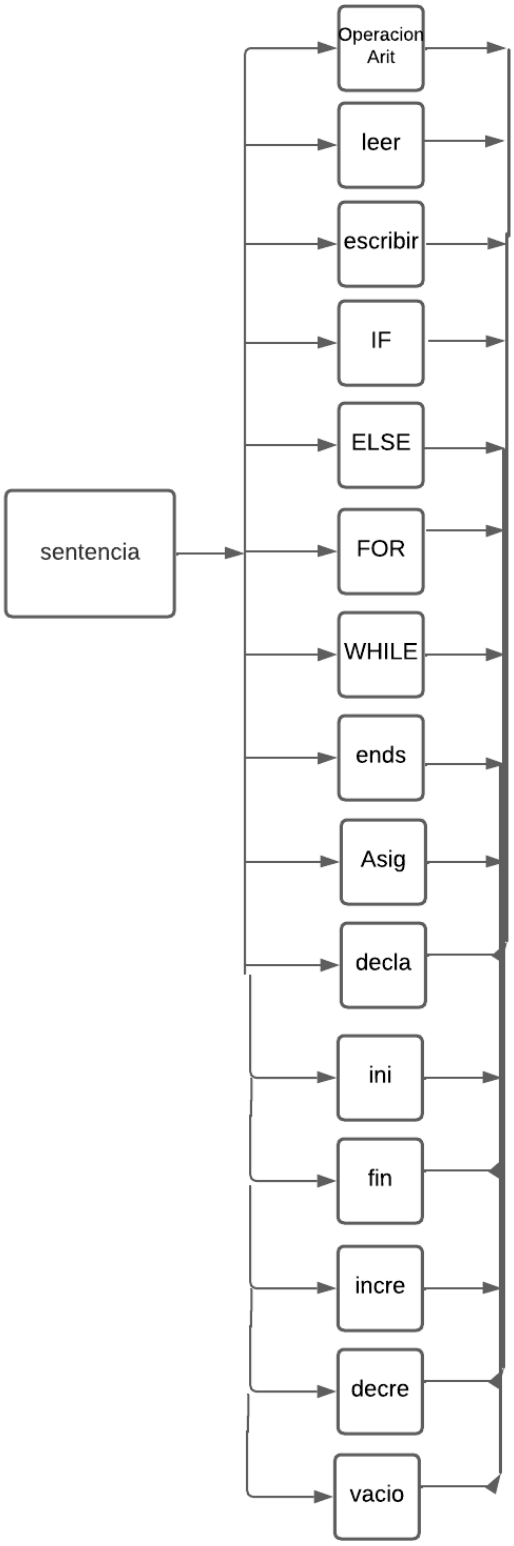

ends
eif
efor
eelse
efor


if
(
condicion
)


ELSE
else


FOR
for
(
Asig
condicionSimpleFor
decre
incre
)


WHILE
while
(
condicion
)


sentencia
Operacion Arit
leer
escribir
IF
ELSE
FOR
WHILE
ends
Asig
decla
ini
fin
incre
decre
vacio

¿Qué es la pila semántica?

La "pila semántica" se refiere a un concepto en informática y programación que se utiliza en la ejecución de programas. Es un área de la memoria que se organiza como una estructura de datos tipo pila (LIFO, Last In, First Out) y se utiliza para almacenar información sobre las operaciones y llamadas a funciones que se están realizando en un programa.

Cuando un programa ejecuta una función o realiza una operación, la información relacionada con esa operación se coloca en la pila semántica. Esto incluye los parámetros de la función, las variables locales y cualquier otra información relevante. A medida que se completan las operaciones, la información se retira de la pila en el orden inverso al que fue colocada.

La pila semántica es esencial para el control de flujo y la gestión de memoria durante la ejecución de programas. Permite que el programa lleve un seguimiento de las llamadas a funciones, maneje los valores de retorno, y gestione las variables locales de manera eficiente. En resumen, es una estructura de datos crucial para el correcto funcionamiento de los programas, especialmente en entornos de programación de bajo nivel como ensamblador y en la implementación de lenguajes de programación.

¿Cómo funciona?

La pila semántica funciona como una estructura de datos que sigue el principio Last In, First Out (LIFO), que significa que el último elemento que se coloca en la pila es el primero en ser retirado. Aquí hay una descripción básica de cómo funciona:

1. **Push (Empujar):** Cuando una función se llama o una operación se ejecuta, la información relevante, como los parámetros de la función y las variables locales, se coloca en la pila. Este proceso se llama "empujar" o "push".

2. **Pop (Sacar):** Cuando se completa la función o la operación, la información asociada a esa operación se retira de la pila. Este proceso se llama "sacar" o "pop". La información se retira en el orden inverso al que fue colocada, ya que el último elemento agregado es el primero en ser retirado.

3. **Gestión del Puntero de Pila:** Un puntero de pila, también conocido como puntero de pila semántica, se utiliza para realizar un seguimiento del extremo superior de la pila, indicando dónde se agregarán o retirarán los elementos. Cuando se realiza un "push", el puntero se mueve hacia arriba; cuando se realiza un "pop", el puntero se mueve hacia abajo.

4. **Alcance de las Variables Locales**: La pila semántica también se utiliza para gestionar el alcance de las variables locales. Cuando se entra en una función, se pueden agregar las variables locales a la pila, y cuando se sale de la función, se eliminan de la pila.

En resumen, la pila semántica proporciona una forma eficiente de gestionar el flujo de ejecución y la memoria durante la ejecución de programas. Permite mantener un rastro de las llamadas a funciones, manejar los valores de retorno y gestionar las variables locales de manera ordenada. La estructura LIFO facilita la gestión de las operaciones en el orden en que se ejecutan.

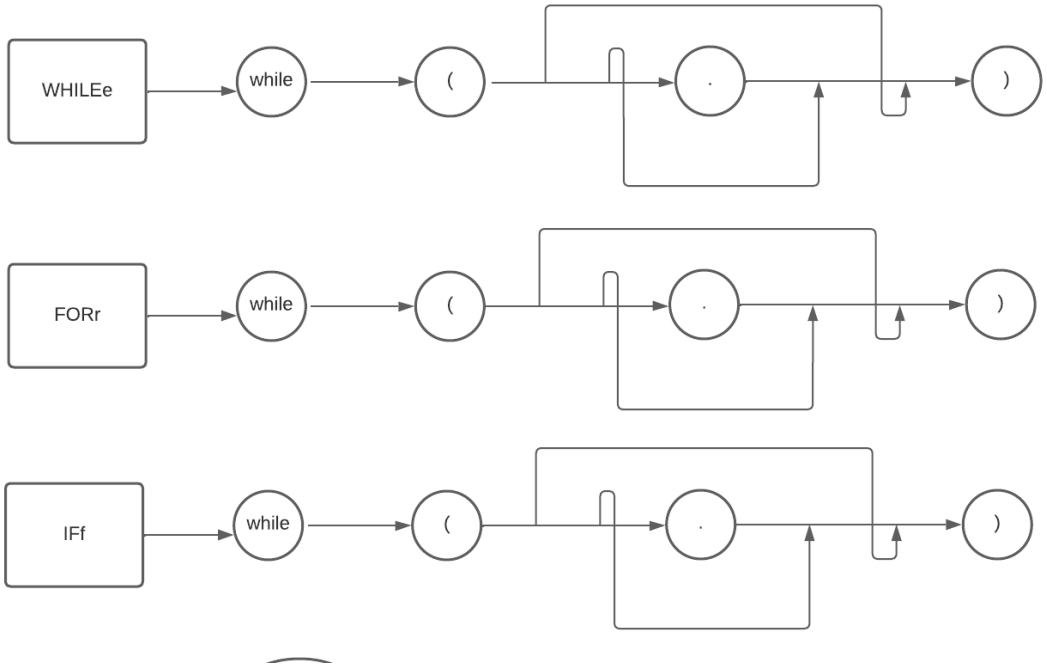
Acciones semánticas y errores semánticos por sentencia

Sentencia	Lo que acepta	Acciones semánticas	Errores semánticos	Acciones secundarias
Declaración	<tipo> <ID> ; <tipo>:: int|string|float	Verificar que los tipos sean validos	Tipos no validos	Error en la lista y seguir con ejecución
Asignación	<Asig>:: <ID> = <operando> <operando>:: <ID>|<constante>	Verificar que el tipo del operando conicida con el ID al que se le va a asignar y que el id se encuentre en la tabla de simbolos	Tipos no coinciden y la línea Varibale no declarada	Error en la lista y seguir con ejecución
Constante	<constante>:: <CteInt>|<CteFloat> | <CteString>	Verificar que el dato ingresado sea alguno de las constantes disponibles	Constante no valida	Error en la lista y seguir con ejecución
Operación Aritmética	<OperacionArit>:: <Asig> <OpAritmetico> <operando>	Verificar que el tipo de el operando sea igual al de la asignación y que ningúno de ellos puede ser de tipo string a menos que el operador aritmético sea un +	Tipos no coinciden y la línea String no validos	Error en la lista y seguir con ejecución
Condición	<condicionSimple>:: <operando> <OpRelacional> <operando>	Verificar que los operandos sean del mismo tipo, int o float	Tipos no coinciden y la línea, string no validos	Error en la lista y seguir con ejecución
Leer	<leer>:: leer <ID>;	Verficar que solo se lean ID's	Lectura invalida,	Error en la lista y seguir con la ejecucion

If	<if>:: if (<condicion>)	Hacer acciones semánticas de la condición y evaluar que se abra y cierre	Tipos de operandos incompatibles. Bloque no cerrado. Bloque no abierto	Agregar error a la lista y seguir con la ejecución
for	<for>:: for (<Asig>; <condicionSimpleFor>; <incre>|<decre>)	Hacer acciones semánticas de la condición que son que la asig y la operación deben ser entreras pero la condición puede ser del tipo que sea y evaluar que se abra y cierre	Tipos de operandos incompatibles. Bloque no cerrado. Bloque no abierto	Agregar error a la lista y seguir con la ejecución
while	<while>:: while (<condicion>)	Hacer acciones semánticas de la condición y evaluar que se abra y cierre	Tipos de operandos incompatibles. Bloque no cerrado. Bloque no abierto	Agregar error a la lista y seguir con la ejecución

NOTAS

- Se necesitan poner las etiquetas de inicio y fin de programa para que pueda funcionar
- Las declaraciones de variables que se hagan se toman como globales.
- Se necesita que las variables estén declaradas para poder usarlas en líneas posteriores.
- Se usarán las siguientes expresiones para el buen funcionamiento de la pila semántica, aunque el interior de estas sentencias, no estén bien escritas, se mete a la pila o se saca.


WHILEe
while
(
.
)
FORr
while
(
.
)
IFf
while
(
.
)