



TC3048 Diseño de Compiladores
Omedetou
6 de Junio del 2022

	
Sergio Guasso Idalias	Alejandro Cedillo Gamez

Índice

Descripción y documentación técnica del Proyecto	5
Descripción del Proyecto:	5
Alcance del proyecto	5
Análisis de Requerimientos	5
Descripción de los principales Test Cases	5
Proceso general	7
Bitácora	7
Lista de commitments	8
Reflexiones:	13
Descripción del lenguaje	14
Nombre del lenguaje	14
Características del lenguaje	14
Listado de errores	15
Descripción del compilador	15
Equipo de computo, lenguaje y librerías	15
Descripción de analisis lexico	16
Expresiones Regulares:	16
Listado de Tokens:	21
Descripcion de analisis sintáctico	23
<Program>	23
<Main>	23
<Bloque>	24
<Estatuto>	25
<Asignacion>	25
<Condicion>	26
<Tipo>	26
<Ciclo>	27
<Escritura>	27
<Lectura>	27
<Vars>	28
<Acceder_array>	28
<Expresion>	29
<Exp>	29
<Termino>	30
<Factor>	30
<Class>	31
<Function>	31
<FunctionCall>	32
<Var Cte>	32
Descripción análisis de código intermedio y análisis semántico	33
Direcciones virtuales	33
Rangos Globales	33

Rangos Temporales	33
Rangos locales	33
Rangos Constantes	33
Rango Pointer	34
Puntos neurálgicos	34
<Programa>	34
<Main>	34
<Bloque>	35
<Estatuto>	35
<Asignación>	36
<Condicion>	36
<Tipo>	36
<Ciclo>	37
<Escritura>	37
<Lectura>	38
<Vars>	38
<Expresion>	39
<Exp>	39
<Término>	39
<Factor>	40
<Function>	40
<FunctionCall>	40
<Var Cte>	41
Tabla de consideraciones semánticas	41
Administración de memoria en compilación	41
Myglobal Vars:	41
Diccionario de funciones:	42
Diccionario de constantes:	42
Clase Funciones:	42
Diccionario de objetos:	43
Clase Objetos:	43
PilaO:	43
Popper:	43
PSaltos:	44
Cuádruplos:	44
Máquina Virtual	44
Especificaciones técnicas	44
Pruebas de funcionalidad del lenguaje	45
Documentación del código del proyecto	51
Manual de usuario	58
Guía Rápida	58
Declaración	58
Asignación	58
Condición	58

Impresión	59
Read	59
Ciclo	59
Función	60
Llamada de función	60
Comentario	61
Declarar Clase	61
Video: Manual de usuario	61

Descripción y documentación técnica del Proyecto

Descripción del Proyecto:

Link del repositorio:

[Omedetou Compilador](#)

Alcance del proyecto

Crear un lenguaje que cumpla con todas las reglas gramaticales necesarias para su funcionamiento básico, el cuál será orientado a objetos.

Y adicionalmente se tendrán arreglos en funciones así como en objetos.

El desarrollo del proyecto será durante lo abarcado en el semestre de Feb-Junio 2022

Análisis de Requerimientos

Correcto funcionamiento al llamado de funciones

Correcto funcionamiento al acceso de atributos de objetos

Validación de errores (Ver apartado de validación de errores)

Manejo de inputs y outputs

Descripción de los principales Test Cases

Para ver los archivos ver la carpeta /Tests dentro de el repositorio de Omedetou

Nombre del Test	test_doc_impresion.txt
Propósito	Correcta impresión de constantes

Nombre del Test	test_doc_asignacion.txt
Propósito	Correcta asignación y guardado en memoria de variables

Nombre del Test	test_doc_condicion.txt
Propósito	Manejo correcto de estatutos condicionales

Nombre del Test	test_doc_lectura.txt
Propósito	Correcto manejo de lectura en el sistema

Nombre del Test	test_doc_ciclo.txt
Propósito	Manejo correcto de ciclos anidados

Nombre del Test	test_doc_funcion.txt
Propósito	Comportamiento adecuado al cambio de contexto y registro de parámetros en funciones

Nombre del Test	test_fact_iter.txt
Propósito	Correcta obtención del valor factorial de un número, correcto manejo de cambio de contexto y manejo de parámetros

Nombre del Test	test_fact_recur.txt
Propósito	Manejo correcto de parche guadalupano, y memoria en cambio de contexto recursivo

Nombre del Test	test_find.txt
Propósito	Recorrido adecuado de arreglos

Nombre del Test	test_arreglo_obj_operation.txt
Propósito	Correcto manejo de atributos de objetos y sus funciones

Nombre del Test	test_arreglo_fact.txt
Propósito	Manejo de recursividad con acceso de arreglos

Nombre del Test	test_fib_uno.txt
-----------------	------------------

Propósito	Manejo correcto de condicionales y obtención de fibonacci a x número de forma iterativa
-----------	---

Nombre del Test	test_fib_rec.txt
Propósito	Manejo correcto de condicionales y obtención de fibonacci a x número de forma recursiva

Nombre del Test	test_sort.txt
Propósito	Ordenar de menor a mayor un arreglo

Proceso general

Bitácora

✓ Fase 0

Propuesta de compilador con lista de tokens, diagramas de sintaxis y otras consideraciones.

- Diagramas
- Propuesta inicial

✓ Fase 1

Ejemplo de código en lenguaje Omedetou, archivo sencillo de python con sintaxis en lark y ejemplo de funcionamiento en Lark IDE.

✓ Fase 2

Ejemplo de clase funcional, corrección de diagramas de flujo, e inicio de lógica en puntos neurálgicos.

✓ Fase 3

Semántica básica de expresiones, cubo semántico, generación de expresiones aritméticas y secuenciales.

✓ Fase 4

Generación de código de estatutos condicionales

- Validación de goto
- Validación de ciclos anidados

✓ Fase 5

Generación de código de funciones

- Directorio de funciones
- Creación de objeto clase
- Variables locales
- Parche Guadalupano (Declaración de función en globalVars)
- Recursión

✓ Fase 6

Mapa de memoria / Máquina virtual

- Estatutos secuenciales en MV
- Expresiones aritméticas en MV
- Creación de directorio de objetos
- Creación de objeto para linkar Tabla de Variables y directorio de funciones en objetos
- Mapa de memoria

✓ Fase 7

Máquina virtual

- Funciones lineales
- Validación división en 0
- Corrección de expresiones
- Corrección de parche guadalupano

Compilador

- Correcciones en recursividad

✓ Fase 8

Compilador

- Creación de objetos
- creación de arreglos
- Creación de memoria de pointers
- Corrección en mapa de memoria
- Limpiar temporales
- Funciones recursivas

Documentación:

- Especificaciones de MV
- Especificaciones de Compilador
- Alcance de proyecto
- Corrección de diagramas
- Características del lenguaje

Lista de commitments

[Historial de github](#)

Commits on Apr 13, 2022
<div>Correccion lark.py Guasso committed on Apr 13</div> <div>a4d7c59</div>
<div>Actualizacion Fases iniciales alexcega committed on Apr 13</div> <div>Verified3e3d755</div>
<div>Actualizacion de participantes Guasso committed on Apr 13</div> <div>Verifiedc9364d2</div>
<div>Fase 0 y 1 alexcega committed on Apr 13</div> <div>Verifiedddc6764</div>
<div>Initial commit alexcega committed on Apr 13</div> <div>Verified8cf0b5b</div>
Commits on Apr 26, 2022
<div>correccion RE en DOT alex.cg committed on Apr 26</div> <div>bba7daa</div>
<div>Actualizacion de Tokens Guasso committed on Apr 26</div> <div>29f906a</div>
Commits on Apr 28, 2022
<div>Fase2 Guasso committed on Apr 28</div> <div>a37d750</div>
Commits on Apr 27, 2022
<div>Correccion asignacion de objetos y ejemplo de patos alex.cg committed on Apr 27</div> <div>f559c79</div>
Commits on Apr 30, 2022
<div>typo alexcega committed on Apr 30</div> <div>Verified1b9a39f</div>
Commits on Apr 29, 2022
<div>Merge branch 'main' of https://github.com/alexcega/Omedetou-Compilador alex.cg committed on Apr 29</div> <div>b3cf245</div>
<div>2 ejemplos nuevos alex.cg committed on Apr 29</div> <div>244608d</div>
Commits on May 1, 2022
<div>Merge branch 'main' of https://github.com/alexcega/Omedetou-Compilador alex.cg committed on May 1</div> <div>a40fbb7</div>
<div>puntos condicion, ciclo, print alex.cg committed on May 1</div> <div>dfaeee7</div>
Commits on May 1, 2022
<div>Merge branch 'main' of https://github.com/alexcega/Omedetou-Compilador Guasso committed on May 1</div> <div>64494f7</div>
<div>Actualizacion de propuesta Guasso committed on May 1</div> <div>90e8df1</div>

Merge branch 'main' of https://github.com/alexcega/Omedetou-Compilador		c84ea51	<
AND y OR en lista de tokens		a2cdd57	<
generacion de cubo semantico		7125c80	<
codigo ciclo y condicion		179dc5	<

Commits on May 3, 2022			
librerias	Verified	3a3454a	<
actualizacion py		9a702ab	<
correccion update tokens		f68839e	<
Rename of tokens		80f9d17	<
Experimentacion lark y token coment		26ef5a8	<

Commits on May 4, 2022			
limpieza funciones		c8fe9d8	<
experimentos		28b7498	<

Commits on May 5, 2022			
primer cuadruplo de asignacion string		b13686f	<>
ajuste de expresiones regulares en tokens		9a1326e	<>
correction 2		8e889dc	<>
correccion var_cte		555a1cc	<>
Avance de cubo semantico		3889075	<>
Acceso a tuplas		819e9f8	<>
Avance Cuadruplo suma		1024b21	<>
primer cuadruplo print		6ee35d7	<>
orden de test y pruebas visitor		1534bb3	<>

Commits on May 7, 2022			
comentarios de transformer		6c37494	<
Commits on May 6, 2022			
all tokens		a765930	<
new transformer		e6aba20	<

Commits on May 9, 2022		
Limpieza de codigo general, ...	alexcega committed 22 days ago	f6593b0 <>
Operadores de comparacion	Guasso committed 22 days ago	b0c8366 <>
correccion asociatividad izquierda	alexcega committed 22 days ago	22e6e2b <>
Commits on May 8, 2022		
cuadрупos if y while ...	alex.cg committed 23 days ago	b83267a <>
cambio gramatica identificador bool	alex.cg committed 23 days ago	d8a33b1 <>
Commits on May 11, 2022		
asignacion y declaracion ...	alexcega committed 20 days ago	db8a63e <>
Correccion tokens asignacion y vars, codigo de asignacion y vars	alexcega committed 20 days ago	57ac133 <>
Limpieza de codigo, implementacion de cubo semantico ...	alexcega committed 21 days ago	f7ff452 <>
Commits on May 10, 2022		
correccion np de print	alexcega committed 21 days ago	1f2f237 <>
cuadрупos repetidos dentro de if	alexcega committed 21 days ago	23e18f1 <>
Commits on May 15, 2022		
creacion test case de tarea	alexcega committed 16 days ago	232ed8c <>
NP de funciones, formato de test cases	alexcega committed 16 days ago	d47e223 <>
Commits on May 13, 2022		
Posibles implementaciones de funciones	alexcega committed 18 days ago	355746f <>
Commits on May 16, 2022		
Fase 5	alexcega committed 15 days ago	Verified 6cfdcbc <>
avance funciones	alexcega committed 15 days ago	0f19d51 <>
Correccion de gramatica	Guasso committed 15 days ago	72514b3 <>
Correccion Diccionario	Guasso committed 15 days ago	384e0b9 <>
Convertidor MV	Guasso committed 15 days ago	926f25e <>
Commits on May 18, 2022		
Merge branch 'main' of https://github.com/alexcega/Omedetou-Compilador	alexcega committed 13 days ago	20b5a63 <>
Avance correcciones en Funciones de memoria local ...	alexcega committed 13 days ago	db59d6d <>

Commits on May 21, 2022		
Correccion de memoria local en funciones ...	alexcega committed 10 days ago	5c9e308 <>
asignacion tareas semana 6	alexcega committed 10 days ago	00a31cb <>
Commits on May 24, 2022		
Avances fase 6	alexcega committed 7 days ago	4055df4 <>
Fase 6	alexcega committed 7 days ago	Verified a0a3335 <>
Commits on May 22, 2022		
avance print atributo de objeto	alexcega committed 9 days ago	cdc93b <>
cambio de mensajes de errores	alexcega committed 9 days ago	ad5a630 <>
Commits on May 26, 2022		
MV if else complete	alexcega committed 5 days ago	064ca63 <>
Commits on May 25, 2022		
avance direcciones de memoria funciones	alexcega committed 6 days ago	90242af <>
Creacion MV, Manejo de Memoria	alexcega committed 7 days ago	2baaac9 <>
Commits on May 28, 2022		
Correccion cantidad de parametros	alexcega committed 3 days ago	16195be <>
MV Read	alexcega committed 3 days ago	Feb4a66 <>
Reasignacion de valores ...	alexcega committed 3 days ago	b9d8cb7 <>
Commits on May 29, 2022		
Mv ejecucion de funciones lineales	alexcega committed 2 days ago	e47f618 <>
Commits on May 30, 2022		
Parametros en funciones de objetos	alexcega committed 22 hours ago	bdc9418 <>
cprreccion memoria temporal en arreglos	alexcega committed 23 hours ago	02015f2 <>
test case objeto ...	alexcega committed 23 hours ago	b6060ae <>
Acceso a variables de objetos ...	alexcega committed 23 hours ago	6c0f065 <>
Avance arreglos	Guasso committed yesterday	2dd8c4e <>

Commits on Jun 1, 2022		
Factorial iterativo	alexcega committed 2 days ago	7862d1e
Operaciones de arreglos	alexcega committed 2 days ago	faf8eb2
Correccion direccion memoria float	alexcega committed 2 days ago	eFa6d3b
suma con matriz	alexcega committed 2 days ago	deadb1a
Reseteo correccion de objeto	alexcega committed 3 days ago	39e7a55
Arreglos en funciones, objetos y funciones de objetos	alexcega committed 3 days ago	a0e4bd5
Commits on Jun 2, 2022		
Correccion iterat fact	alexcega committed 23 hours ago	d313033
Pass recursion de 1 nivel	alexcega committed 23 hours ago	5824c7c
resetear contador, correccion Cubo	alexcega committed yesterday	dc5cc3b
Limpieza de comentarios	alexcega committed yesterday	750709d
Commits on Jun 3, 2022		
Factorial recursivo	alexcega committed 2 hours ago	8e6f738
Correccion gramatica	Guasso committed 9 hours ago	510890c
TestCases_Doc	Guasso committed 12 hours ago	3dc0493
Avance_DocumentacionFinal	Guasso committed 14 hours ago	27e33b9
Cubo Semantico	Guasso committed 14 hours ago	971c6b5
Maquina_virtual_recursion	Guasso committed 14 hours ago	a0312c8
Commits on Jun 4, 2022		
Actualizacion de rangos	alexcega committed now	ce70f7b
Ultimos test Cases	alexcega committed 10 minutes ago	c626b4a
Commits on Jun 7, 2022		
Sort Case, delete of prints	alexcega committed 12 seconds ago	f8988f6
gramatica sort	Guasso committed 7 hours ago	7953a38

Reflexiones:

Alejandro: A lo largo del proyecto tuve varias dudas respecto a la funcionalidad del lenguaje por desarrollar, no entendía las limitaciones de tiempo y de alcance de la materia para el nivel de inteligencia que llegaría el lenguaje, ahora puedo ver con mayor claridad y entender aún mejor porque se le llama a un lenguaje como python un lenguaje de 3ra generación a comparación de c++, del mismo modo me hace apreciar aún más el lenguaje con el que me enseñaron a programar y me hace tenerle aún más cariño ya que una de las cosas que me gustan es lo específico pero no excesivo que tienes que ser al programar en c++.

Hablando más del proyecto puedo decir que ahora entiendo lo que implica que un lenguaje acepte lo que le quieres escribir, no más y no menos ya que cuando realizamos la gramática la ambigüedad podría ser algo que parecía imposible de resolver en su momento, después llegamos a el directorio de funciones y entendí la necesidad de el arbol de parseo, como recorrerlo con la ayuda de la librería lark y conforme avanzaba o iba con compañeros para preguntar alguna duda puede entender la diferencia entre nuestros parsers, considero que ahora con lo que se hay cosas que pude hacer diferentes pero creo que la forma en que lo trabaje sigue siendo la adecuada. Por último considero que la materia no concuerda con la cantidad de unidades que vale ya que el equivalente de horas dedicadas necesarias para realizar el proyecto no coincide con las establecidas (8) en el mapa de carrera.



Sergio: La creación de este proyecto me permitió ver la cantidad de instrucciones que se necesitan para poder hacer que un lenguaje de programación funcione correctamente. Tan solo el crear funciones consideradas básicas requiere una gran comprensión de la gramática necesaria y puntos neurálgicos.

Me ayudó a poner en práctica los temas vistos en clase, y a comprenderlos mejor; me ayudó a practicar más el ámbito de la programación y darle un enfoque diferente de cualquier proyecto que antes había hecho.



Descripción del lenguaje

Nombre del lenguaje

Omedetou

Características del lenguaje

Uso de expresiones lineales tales como:

1. Lectura
2. Asignaciones
3. Expresiones
4. Escritura
5. Declaraciones
6. Reasignaciones
7. Comentarios

Uso de expresiones no lineales tales como:

1. If

2. Else
3. While

Cambios de contexto como:

1. Funciones
2. Funciones de objetos

Tipos de datos como:

1. int
2. bool
3. string
4. float

- La declaración de variables debe seguir el siguiente ejemplo:
Var int numero = 10
Donde puede ser uno de los tipos previamente mencionados y el valor puede no ir o ser una expresión más complicada o una llamada a una función no void.
- Las funciones pueden o no regresar un valor, esto es ser de tipo void o de alguno de los otros tipos mencionados.
- Se permite la asignación de variables globales fuera de funciones así como creación de variables locales en main.
- Soportamos ciclos anidados así como condicionales anidados.

Listado de errores

A continuación se muestra un listado de los errores soportados por el lenguaje Omedetou

1. Variable no declarada
2. Print void function
3. Objeto no declarado
4. Función no definida
5. Syntax error, en condicionales, resultando no bools
6. Type error, en expresiones con ayuda del cubo semántico
7. Zero división error
8. Error de doble declaración de variable
9. Error de doble declaración de objeto
10. Error de doble declaracion de funcion
11. Error de parámetro duplicado (Nombre)
12. Error de tipo de parámetro en llamada de función
13. Error en menor cantidad llamada de parámetros
14. error en mayor cantidad llamada de parámetros
15. Error de conversion en Read
16. Index error en array
17. Elemento de array no definido

De igual forma se recomienda ver el archivo validationErrors.py

Descripción del compilador

Equipo de computo, lenguaje y librerías

Equipo de computo:

Alex:

Device name LAPTOP-CJITDPM
Processor AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz
Installed RAM 8.00 GB (7.37 GB usable)
Device ID FFBFB4EE-40E7-45EF-85F6-49DAECFC9C28
Product ID 00325-82074-22926-AAOEM
System type 64-bit operating system, x64-based processor
Pen and touch No pen or touch input is available for this display

Edition Windows 10 Home

Version 21H1
Installed on 6/2/2021
OS build 19043.1706
Serial number PF2M0539
Experience Windows Feature Experience Pack 120.2212.4170.0

Guasso:

Device name DESKTOP-V73E29T
Processor Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz
Installed RAM 16.0 GB (15.8 GB usable)
Device ID 46C1A479-3E21-46FF-972B-9210DE53DEB6
Product ID 00325-95827-25195-AAOEM
System type 64-bit operating system, x64-based processor
Pen and touch Pen and touch support with 10 touch points

Edition Windows 10 Home

Version 21H2
Installed on 3/14/2021
OS build 19044.1706
Experience Windows Feature Experience Pack 120.2212.4170.0

Lenguaje:

Python 3.9.5 64bits

Librerías:

- Lark
- Collections
- copy

Descripción de análisis lexico

Expresiones Regulares:

//tokens

START_PROGRAM : "Start"

FINISH_PROGRAM : "Finish"

VAR : "Var"

WHILE : "While"

IF : "If"

ELSE : "Else"

DEF : "Def"

RETURN : "Return"

CLASS : "Class"

PRINT : "Print"

MAIN : "Main"

VOID : "Void"

READ : "Read"

LEFT_CURRLY_BRACES : /\{/

RIGHT_CURRLY_BRACES : /\}/

LEFT_PARENTHESIS : /\(/

RIGHT_PARENTHESIS : /\)/

LEFT_BRACKET : /\[/

RIGHT_BRACKET : /\]/

COMMA : /,/

COLON : /:/

DOT : /\./

NEW_LINE: /\n+/

WHITESPACE: (" " | \t/)+

GREATER_THAN : />/

LESS_THAN : /</

LESS_EQUAL : /<=/

GREATER_EQUAL : />=/

NOT_EQUAL : /!=/

EQUAL_COMPARATION : /==/

EQUAL : /=/

OR : /\|/

AND : /\&/

PLUS : /\+/

MINUS : /\-/

ASTERISK : /*/

```

SLASH      : \/

INT_WORD   : "int"
FLOAT_WORD : "float"
BOOL_WORD  : "bool"
STRING_WORD : "String"

IDENTIFIER: /[a-z_][a-zA-Z0-9_]/
CONST_INT: /\d+/
CONST_FLOAT: /\d+(\.\d+)/
CONST_BOOL: "True" | "False"
CONST_STRING : STRING

COMMENT : /#.* /

%ignore WHITESPACE
%ignore NEW_LINE
%import common.ESCAPED_STRING -> STRING

// rules
start: programa

programa : START_PROGRAM NEW_LINE programa2 FINISH_PROGRAM
programa2 : some_vars* np_fin_igualacion (class*|COMMENT) (function*|COMMENT)
start_main
np_fin_igualacion :
some_vars : vars np_limpiar_temps| asignacion np_limpiar_temps | COMMENT
np_limpiar_temps:

start_main: VOID MAIN LEFT_PARENTHESIS RIGHT_PARENTHESIS bloque
np_fin_funcion
np_fin_funcion:

vars : VAR tipo IDENTIFIER EQUAL np_meter_igual expresion np_asiganar_valor
NEW_LINE -> var_con_valor
      | VAR tipo IDENTIFIER -> var_sin_valor
      | VAR tipo IDENTIFIER LEFT_BRACKET exp np_get_lim_s RIGHT_BRACKET
arreglo1 np_calcular_m np_arr_next_virtualadress NEW_LINE -> var_arreglo
      | VAR IDENTIFIER IDENTIFIER ->
var_objeto

arreglo1 : LEFT_BRACKET np_add_dimension exp RIGHT_BRACKET arreglo2 |
-> arr_param
arreglo2 : LEFT_BRACKET np_add_dimension exp RIGHT_BRACKET |
-> arr_param2
np_asiganar_valor:
np_meter_igual:
np_add_dimension:

```

np_arr_bracket1:
np_arr_dim:
np_arr_bracket2:
np_get_lim_s:
np_calcular_m:
np_arr_next_virtualadress:

asignacion : IDENTIFIER EQUAL np_meter_igual expresion np_asignar_valor
NEW_LINE -> reasignar
| IDENTIFIER DOT IDENTIFIER EQUAL expresion NEW_LINE -> reasignar_obj
| acceder_array EQUAL np_meter_igual expresion np_reasignar_arr ->
reasignar_casilla_matriz

acceder_array : IDENTIFIER LEFT_BRACKET np_ver_dimension_access np_meterff
expresion np_arr_ver_expresion RIGHT_BRACKET acceso_matriz* np_fin_array
np_sacarff
acceso_matriz : np_actualizar_dim LEFT_BRACKET np_ver_dimension_access
np_meterff expresion np_arr_ver_expresion RIGHT_BRACKET

size_arr : IDENTIFIER COLON "size"

np_reasignar_arr:
np_actualizar_dim:
np_ver_dimension_access:
np_arr_ver_expresion:
np_fin_array:

bloque: LEFT_CURRLY_BRACES (b1?|function_return)+ RIGHT_CURRLY_BRACES
b1 : estatuto b1*

estatuto : asignacion -> igualar
| condicion -> unif
| escritura -> unprint
| ciclo -> unwhile
| vars -> unavars
| function -> unafuncion
| function_call -> unallamadafuncion
| read_value -> lectura
| COMMENT -> comentario

read_value: READ LEFT_PARENTHESIS IDENTIFIER RIGHT_PARENTHESIS

escritura : PRINT LEFT_PARENTHESIS expresion esc2* np_print RIGHT_PARENTHESIS
NEW_LINE
np_print:
esc2: COMMA np_print expresion

condicion : IF LEFT_PARENTHESIS expresion RIGHT_PARENTHESIS np_falsoif bloque
 condicionelse? np_finif
 np_falsoif :
 np_finif :

condicionelse : np_inicioelse ELSE bloque
 np_inicioelse :

ciclo : WHILE np_iniciowhile LEFT_PARENTHESIS expresion RIGHT_PARENTHESIS
 np_truewhile bloque np_endwhile NEW_LINE
 np_iniciowhile :
 np_truewhile :
 np_endwhile :

expresion : exp expresion1?
 expresion1 : expresion_comparacion exp+ np_comparacion np_comparacion_andor
 expresion_comparacion : GREATER_THAN -> np_meter_mayorque
 | LESS_THAN -> np_meter_menorque
 | EQUAL_COMPARATION -> np_meter_igual_igual
 | NOT_EQUAL -> np_meter_no_igual
 | GREATER_EQUAL -> np_meter_mayor_igual
 | LESS_EQUAL -> np_meter_menor_igual
 | OR -> np_meter_or
 | AND -> np_meter_and

np_comparacion:
 np_comparacion_andor:

exp : termino np_sumarnumeros ex1?
 np_sumarnumeros:
 ex1 : exp_mas_menos exp
 exp_mas_menos : PLUS -> np_metermas
 | MINUS -> np_metermenos

termino : factor np_multiplicarnumeros t1?
 np_multiplicarnumeros:
 t1 : termino_por_entre termino
 termino_por_entre : ASTERISK -> np_meterpor
 | SLASH -> np_meterentre

factor : factor_fondofalso
 | factor_var
 factor_fondofalso : LEFT_PARENTHESIS np_meterff expresion RIGHT_PARENTHESIS
 np_sacarff
 np_meterff:
 np_sacarff:

factor_var : var_cte -> guardar_cte

| IDENTIFIER DOT IDENTIFIER -> guardar_var_de_obj
| acceder_array -> guardar_acceder_array
| size_arr -> arr_size

class : LESS_THAN CLASS GREATER_THAN IDENTIFIER LEFT_CURRLY_BRACES
np_inicio_vars_obj (vars|COMMENT)* np_fin_vars_obj (function|COMMENT)*
RIGHT_CURRLY_BRACES np_fin_clase->inicializar_clase
np_inicio_vars_obj:
np_fin_vars_obj:
np_fin_clase:

function : DEF (tipo|VOID) IDENTIFIER LEFT_PARENTHESIS function_param?
RIGHT_PARENTHESIS LEFT_CURRLY_BRACES (estatuto* function_return?)+
RIGHT_CURRLY_BRACES np_fin_funcion
function_param : tipo IDENTIFIER is_array? function_params*
function_return : RETURN expresion np_guadalupe
| RETURN LEFT_PARENTHESIS expresion RIGHT_PARENTHESIS
np_guadalupe

np_guadalupe:
function_params : COMMA tipo IDENTIFIER is_array
is_array: LEFT_BRACKET CONST_INT RIGHT_BRACKET

function_call : IDENTIFIER fc_class? LEFT_PARENTHESIS fc_param?
np_reset_count_params RIGHT_PARENTHESIS
fc_class : DOT IDENTIFIER
fc_param : expresion np_check_param np_insert_param fc_params*
np_check_param:
np_insert_param:
np_reset_count_params:
fc_params : COMMA expresion np_check_param np_insert_param

tipo: INT_WORD
| FLOAT_WORD
| STRING_WORD
| BOOL_WORD

var_cte : IDENTIFIER -> identificador
| CONST_INT -> entero
| CONST_FLOAT -> decimal
| CONST_STRING -> palabra
| CONST_BOOL -> booleano
| function_call -> funcion

Listado de Tokens:

Nombre	Signo
START	Start
FINISH	Finish
VAR	Var
WHILE	While
IF	If
ELSE	Else
DEF	Def
RETURN	Return
CLASS	Class
PRINT	Print
MAIN	Main
LEFT_CURRLY_BRACES	{
RIGHT_CURRLY_BRACES	}
LEFT_PARENTHESIS	(
RIGHT_PARENTHESIS)
LEFT_BRACKET	[
RIGHT_BRACKET]
COMMA	,
COLON	:
DOT	.
NEW_LINE	\n
GREATER_THAN	>
LESS_THAN	<
LESS_EQUAL	<=
GREATER_EQUAL	>=
NOT_EQUAL	!=
EQUAL_COMPARATION	==

EQUAL	=
OR	
AND	&
PLUS	+
MINUS	-
ASTERISK	*
SLASH	/
COMMENT	#
VOID	Void
READ	Read
INT_WORD	int
FLOAT_WORD	float
BOOL_WORD	bool
STRING_WORD	String

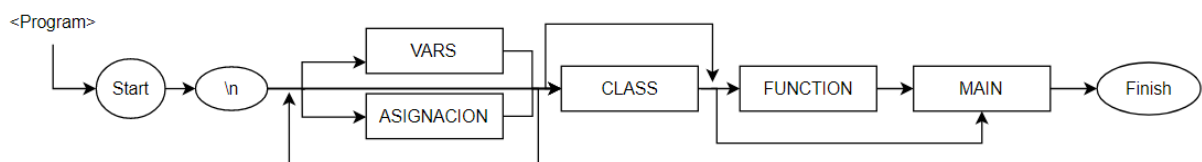
De igual forma se aconseja ver el documento 'tokens omedetou.txt' para mayor apreciación.

Descripcion de analisis sintáctico

A continuación los diagramas del análisis sintáctico, de igual manera pueden verse en el siguiente enlace : [Diagramas](#)

Gramática

<Program>



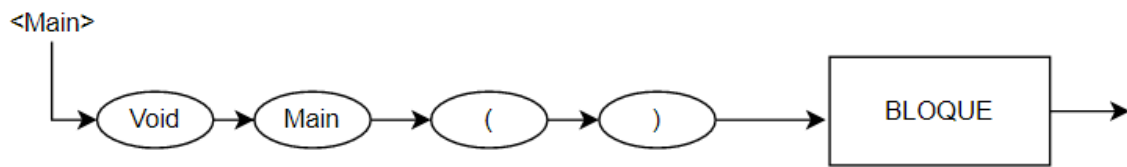
program -> Start \n program2 Finish

program2 -> some_vars* CLASS* FUNCTION* start_main

some_vars -> VARS | ASIGNACIÓN

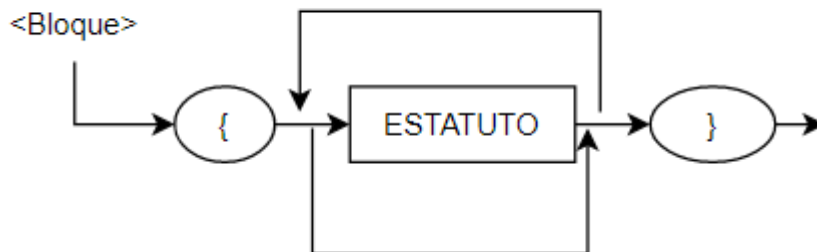
start_main -> MAIN () BLOQUE

<Main>



Main-> Void Main () BLOQUE

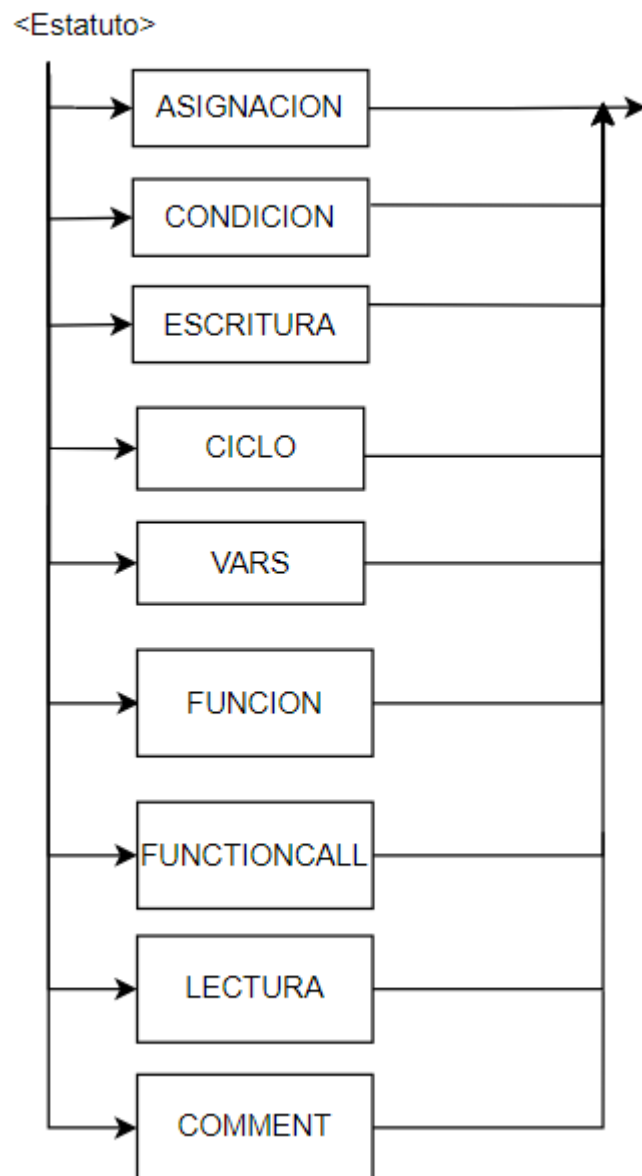
<Bloque>



bloque global -> { b1 }

b1-> ESTATUTO b1

<Estatuto>



Estatuto-> ASIGNACION | CONDICION | ESCRITURA | CICLO | VARS | FUNCTION |
FUNCTION_CALL | LECTURA |COMMENT

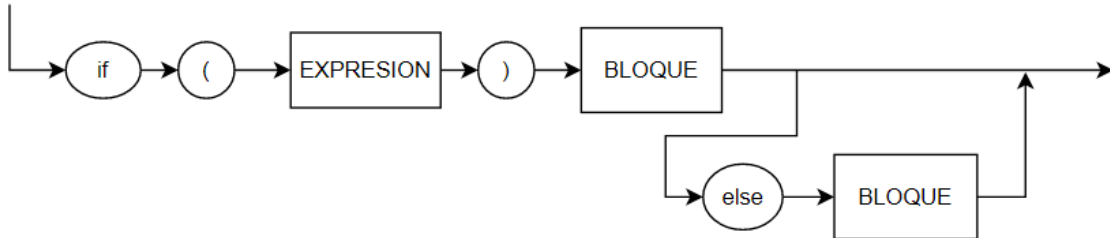
<Asignacion>



asignacion -> id = EXPRESION \n

<Condicion>

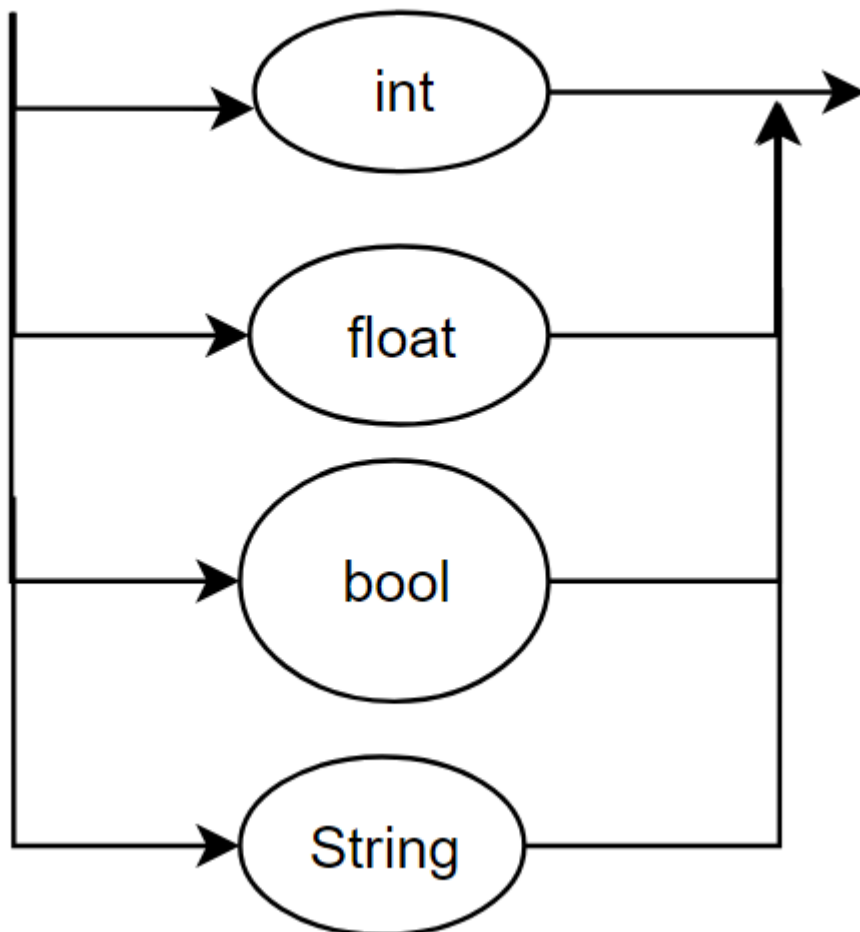
<Condición>



Condicion -> if (EXPRESION) BLOQUE CONDICIONELSE

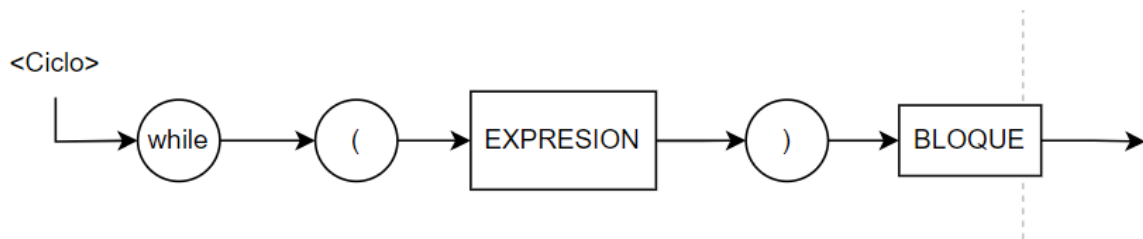
<Tipo>

<Tipo>



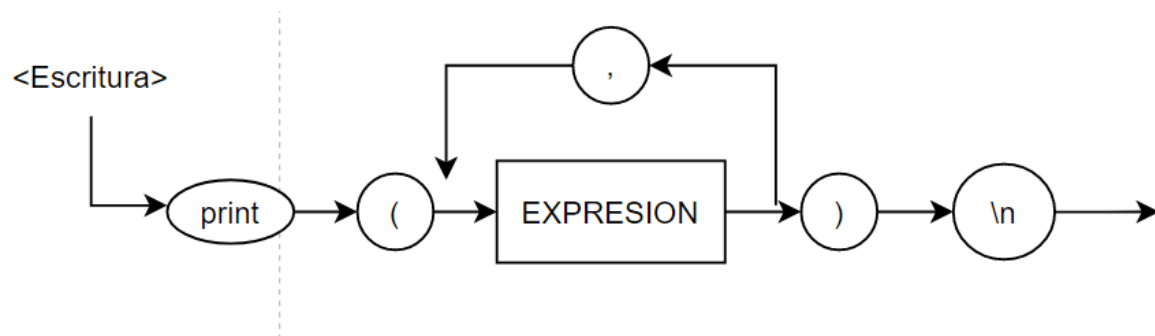
Tipo-> int_word | float_word | bool_word | string_word

<Ciclo>



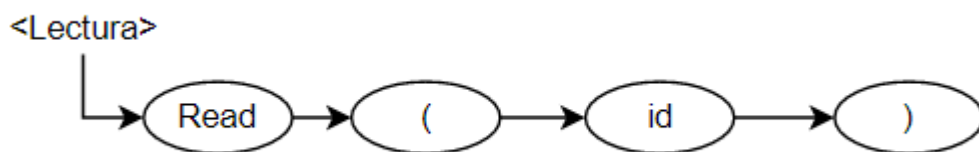
Ciclo-> while (EXPRESION) BLOQUE \n

<Escritura>



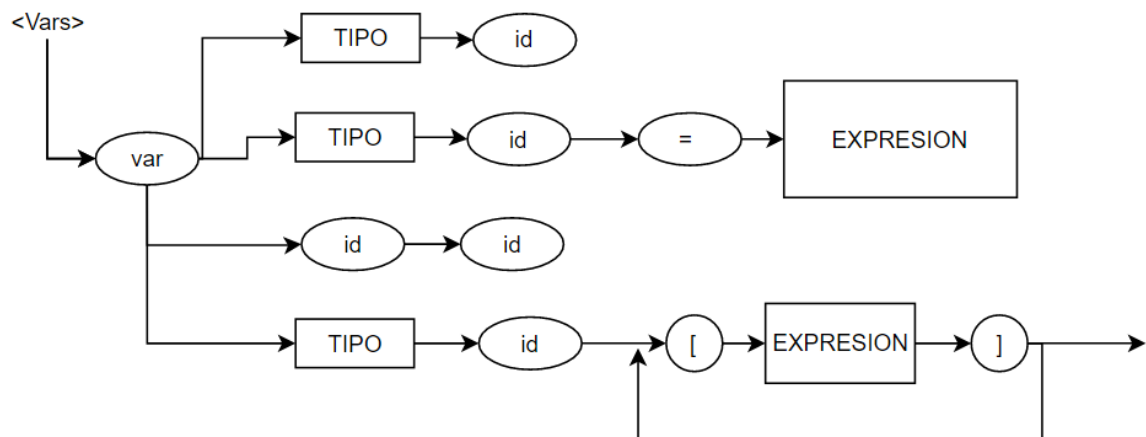
Escritura -> print (EXPRESION ESC2) \n
ESC2 -> , EXPRESION ESC2 |

<Lectura>



Lectura->Read (Id)

<Vars>

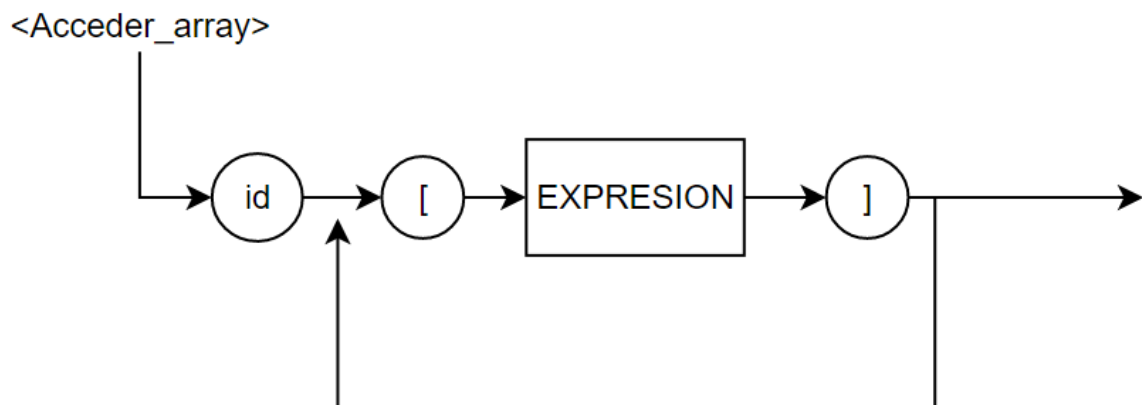


vars-> var TIPO id = EXPRESION | var TIPO id | var id id | var TIPO id [EXPRESION]

masarr

massarr-> [EXPRESION] massarr |

<Acceder_array>

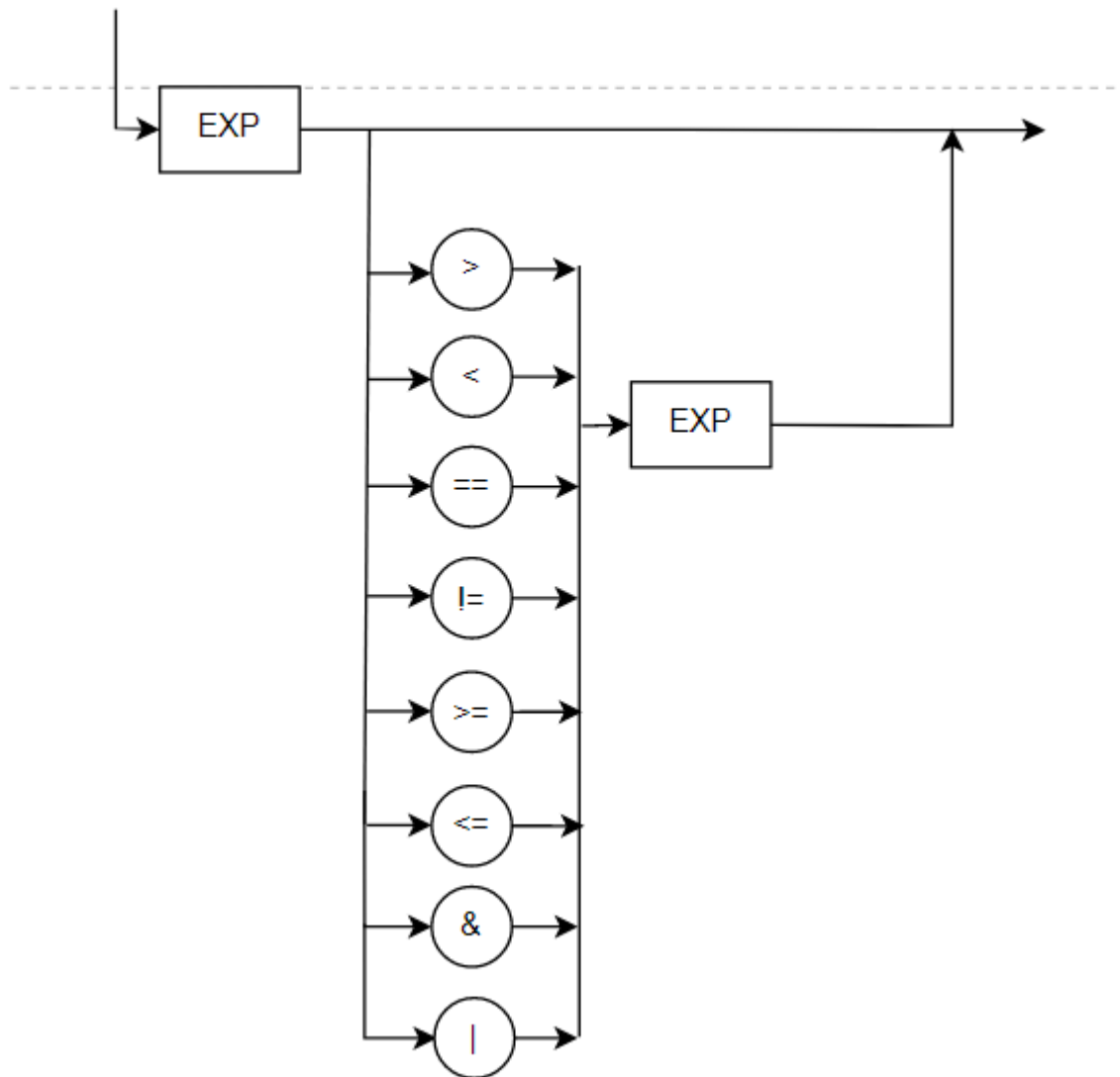


acceder_array-> id [expresion] seguir

seguir-> [expresion] seguir |

<Expresion>

<Expresion>



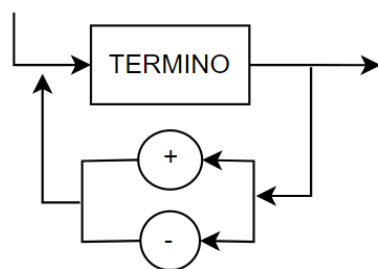
Expresion-> EXP E1

E1 -> EXPRESION_COMPARACION EXP

EXPRESION_COMPARACION -> > | < | == | != | >= | <= | / | &

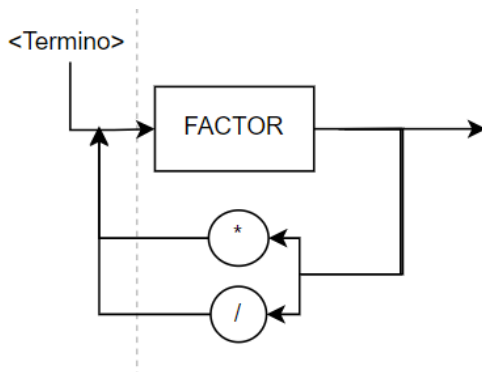
<Exp>

<Exp>



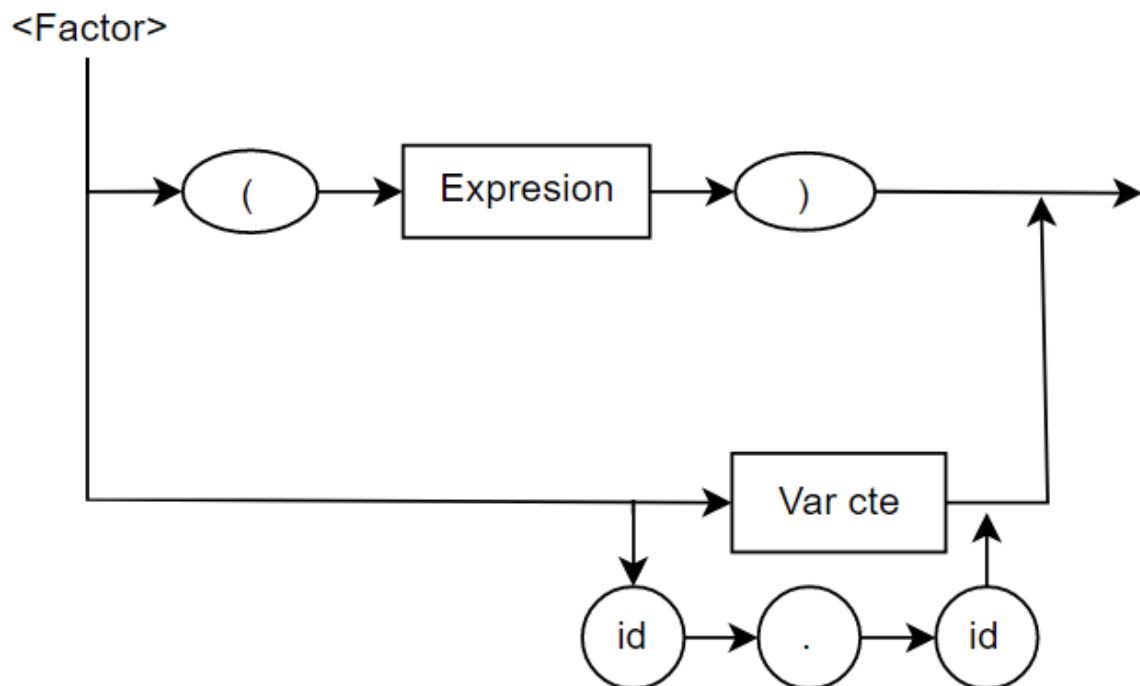
Exp-> TERMINO EXP_MAS_MENOS
 EXP_MAS_MENOS-> + | -

<Termino>



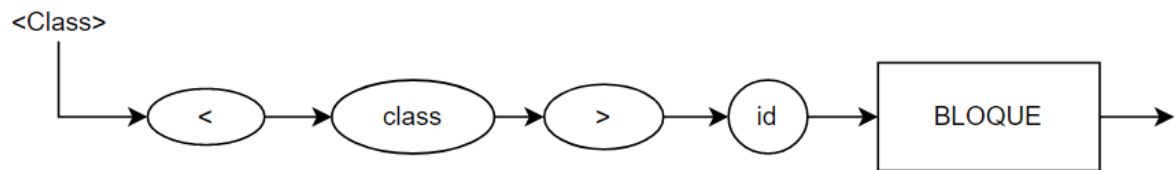
Termino-> FACTOR TERMINO_POR_ENTRE
 TERMINO_POR_ENTRE-> * | /

<Factor>



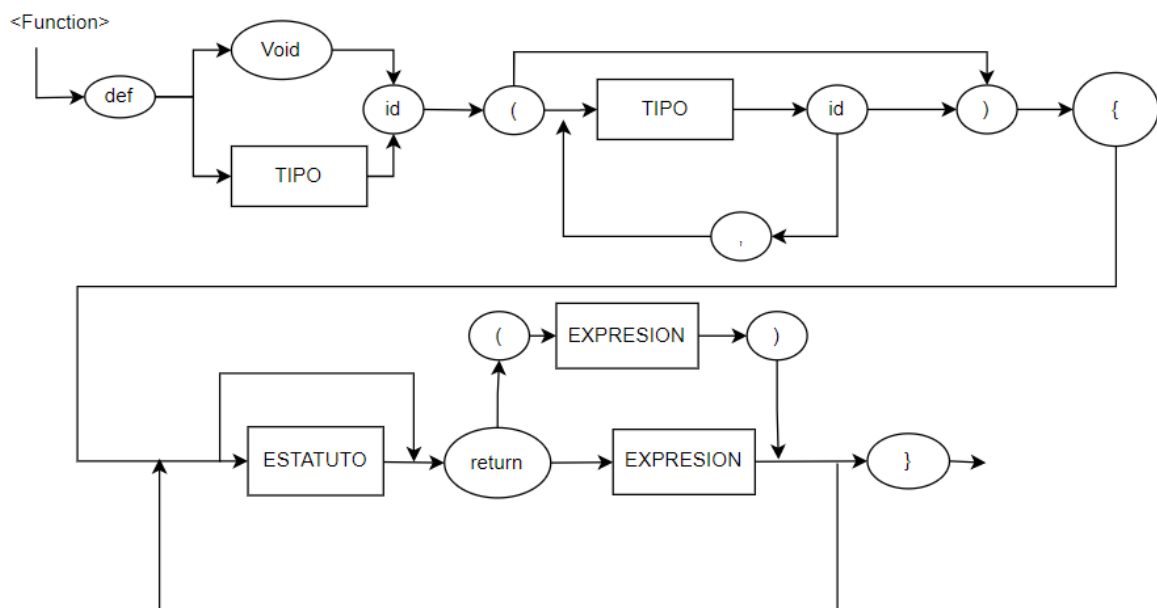
Factor-> factor_fondofalso | factor_var
 factor_fondofalso-> (EXPRESION)
 factor_var-> VAR_CTE | id . id

<Class>



Class-> < class > id BLOQUE

<Function>



Function-> def gettype id (function_param) { contenidoFun }

gettype-> Void | TIPO

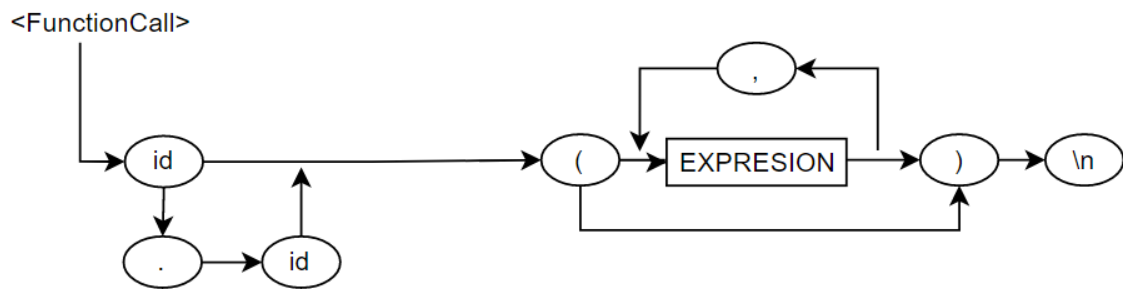
function_param-> ϵ | parametros

parametros -> Tipo VARCTE params2

params2-> ,Tipo VARCTE params2 | ϵ

contenidoFun-> ESTATUTO function_return contenidoFun | ϵ

<FunctionCall>



FunctionCall-> id fc1 (fc2) \n

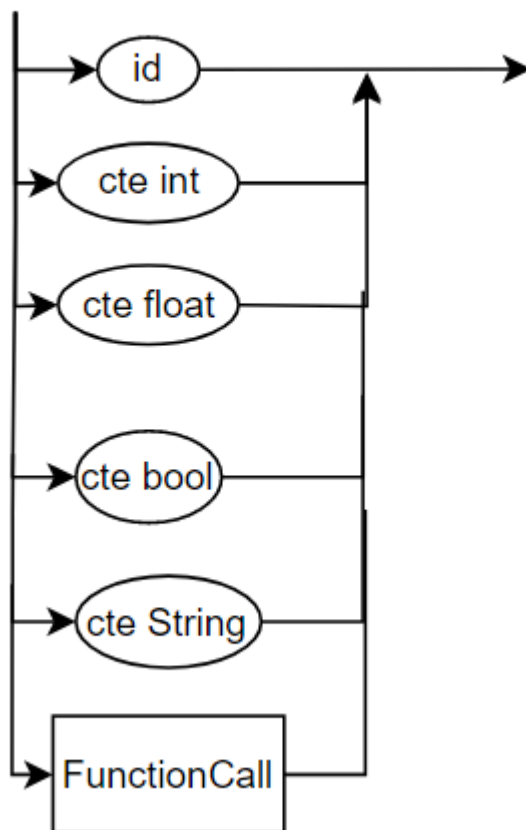
fc1-> . id | ε

fc2-> EXPRESION fc3 | ε

fc3-> , EXPRESION fc3 | ε

<Var Cte>

<Var Cte>



VarCte -> id | cte int | cte float | cte bool | FunctionCall

Descripción análisis de código intermedio y análisis semántico

Direcciones virtuales

Para más información ver el archivo `memoryManager.py`

Rangos Globales

Rango int	1 - 1000
Rango float	1001 - 2000
Rango bool	2001 - 3000
Rango String	3001 - 4000

Rangos Temporales

Rango temp int	4001 - 5000
Rango temp float	5001 - 6000
Rango temp bool	6001 - 7000
Rango temp string	7001 - 8000

Rangos locales

Rango local int	8001 - 9000
Rango local float	9001 - 10000
Rango local bool	10001 - 11000
Rango local string	11001 - 12000

Rangos Constantes

Rango const int	12001 - 13000
Rango const float	13001 - 14000

Rango const bool	140001 - 15000
Rango const string	15001 - 16000

Rango Pointer

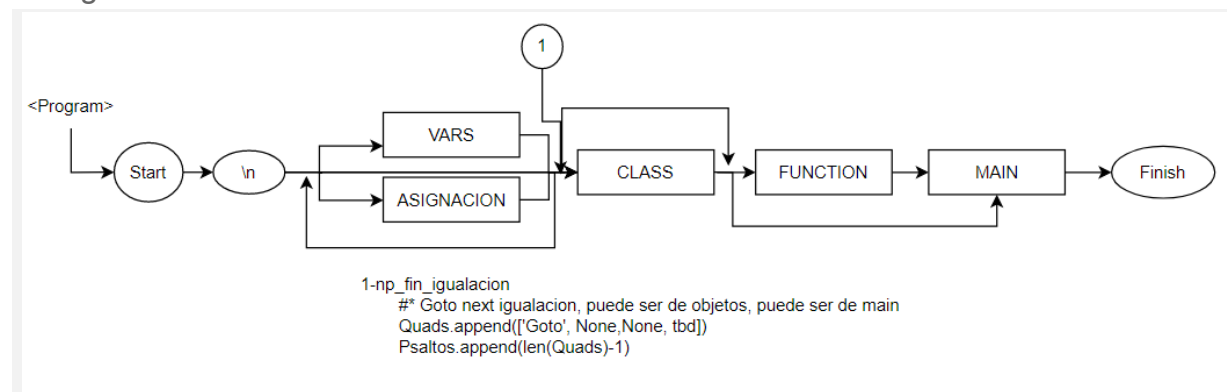
Rango pointer int	16001 - 17000
-------------------	---------------

Puntos neurálgicos

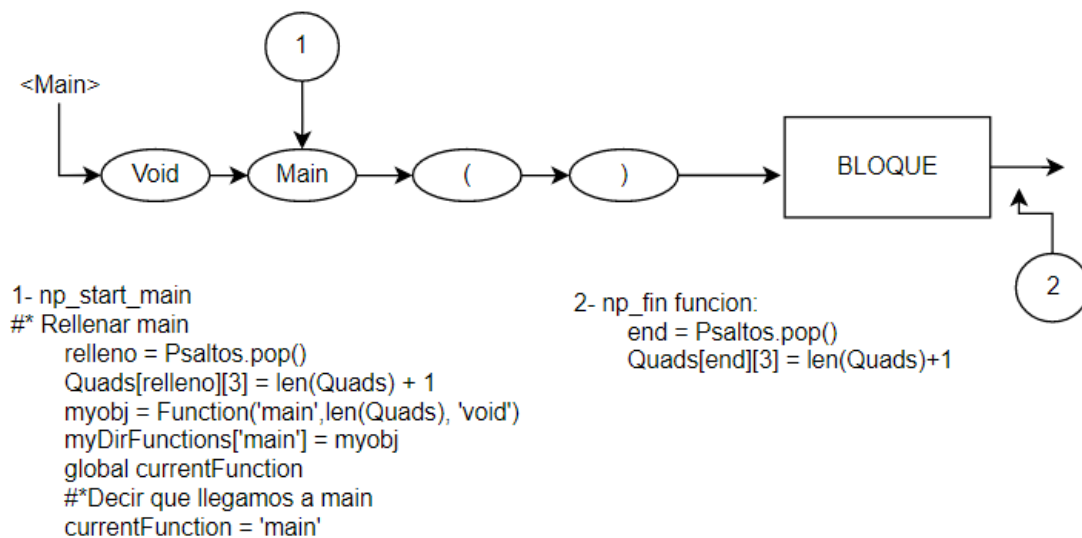
A continuación los diagramas con puntos neurálgicos usados en el proyecto Omedetou, de igual manera pueden verse en el siguiente link:

[Diagramas](#)

<Programa>

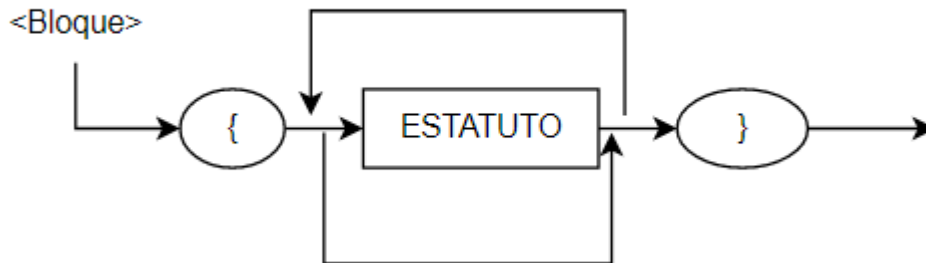


<Main>



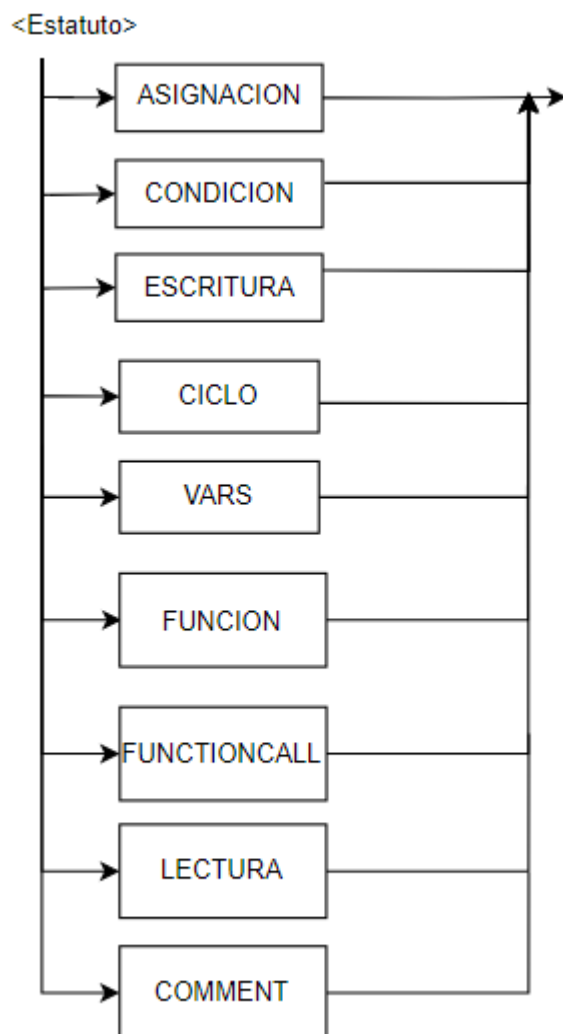
<Bloque>

Bloque no posee puntos neurálgicos

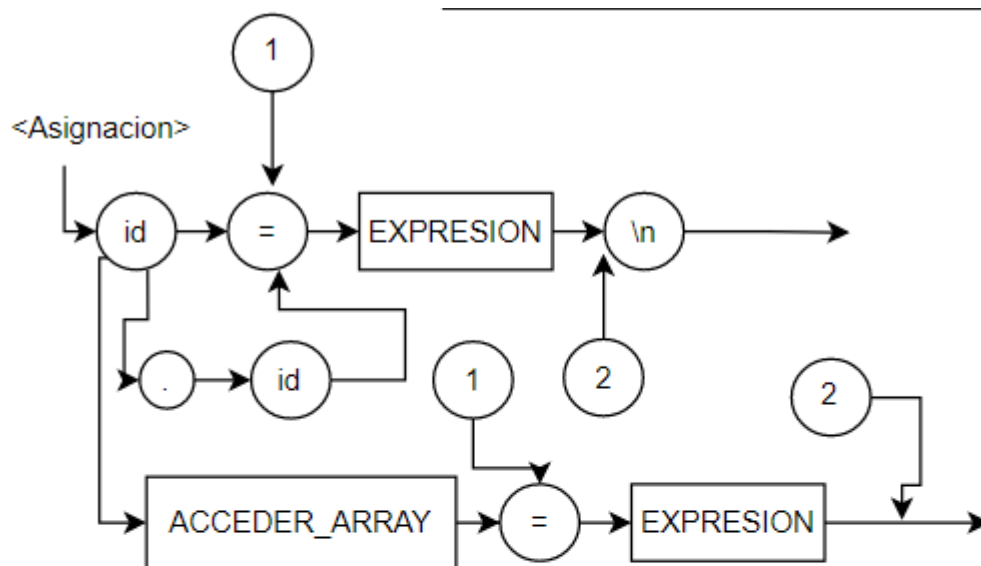


<Estatuto>

Estatuto no posee puntos neurálgicos



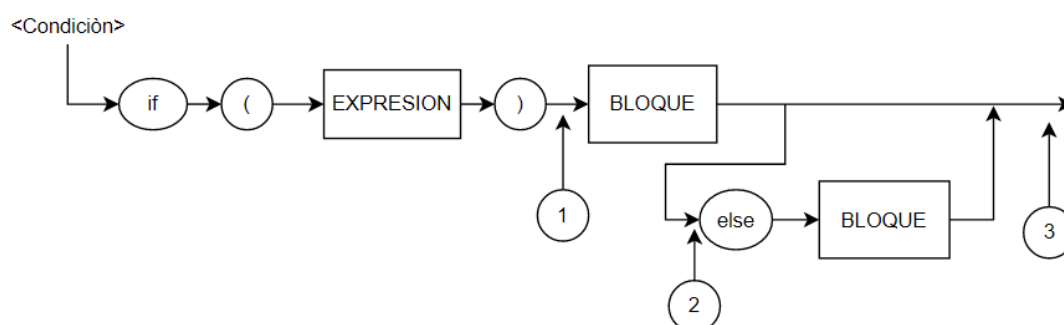
<Asignación>



1- Meter igual en Poper

2- if poper.top() == '=':
 right= pilao.pop()
 left = pilao.pop()
 operador = poper.pop()
 result = getresult(left,right, operador)
 if result != TypeError:
 quads.append([=, right, , left])
 else:
 error type

<Condición>



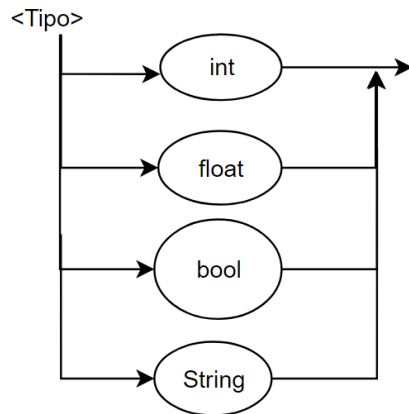
1- falsoif:
 condicion = pilao.pop()
 if condicion.type not bool:
 Error en sintaxis
 else:
 quads.append([gotof, condicion,, tbd])

2- inicioelse:
 quads.append([goto,,tbd])
 false = psaltos.pop()
 psaltos.append(len(quads)-1)
 quads[false[3] = len(quads)+1

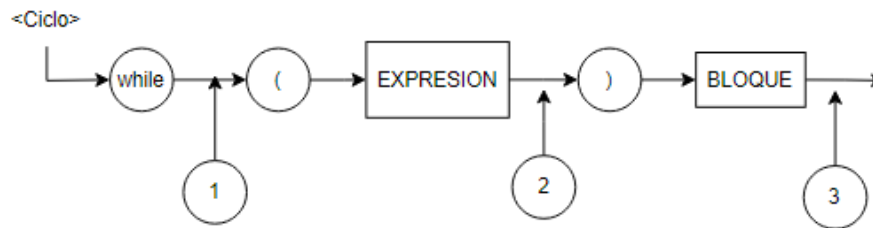
3 - finif :
 end = Psaltos.pop()
 Quads[end][3] = len(Quads)+1

<Tipo>

Tipo no tiene np



<Ciclo>

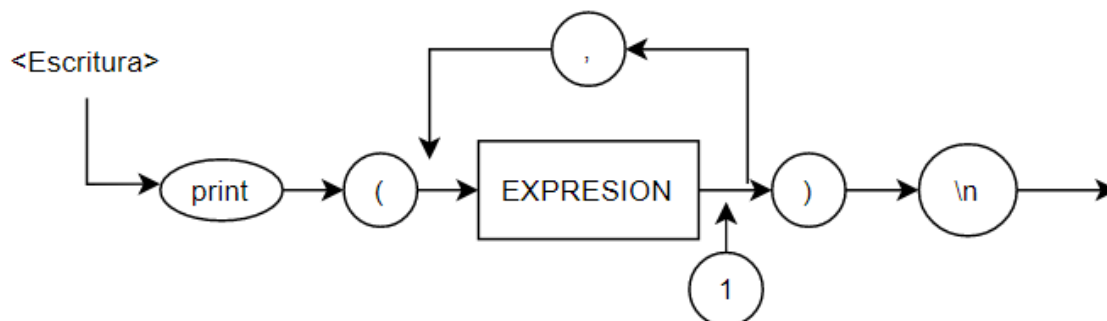


1 - np_iniciowhile
 Psaltos.append(len(Quads)+1)Text

2 - np_truewhile:
 condicion = pilaO.pop()
 #! Error validation
 if condicion['type'] != 'bool':
 print("Syntax Error, expected expression")
 exit()
 else:
 Quads.append(['Goto', condicion['address'], None, tbd])
 Psaltos.append(len(Quads)-1)

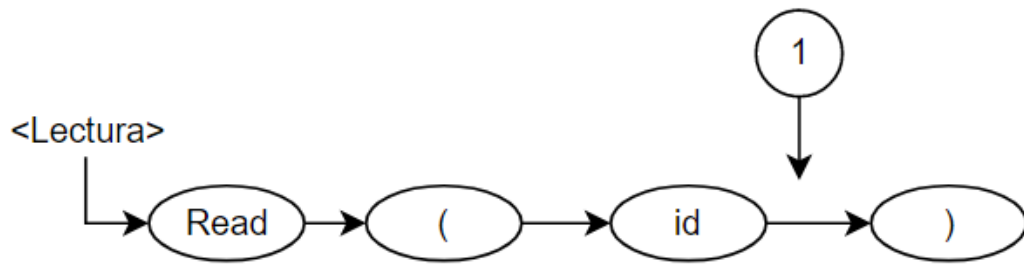
3 - np_endwhile
 falso = Psaltos.pop()
 retorno = Psaltos.pop()
 Quads.append(['Goto', None, None, retorno])
 Quads[falso][3] = len(Quads)+1

<Escritura>



1 - np_print:
 Quads.append(['Print', None, None, pilaO.pop()['address']])

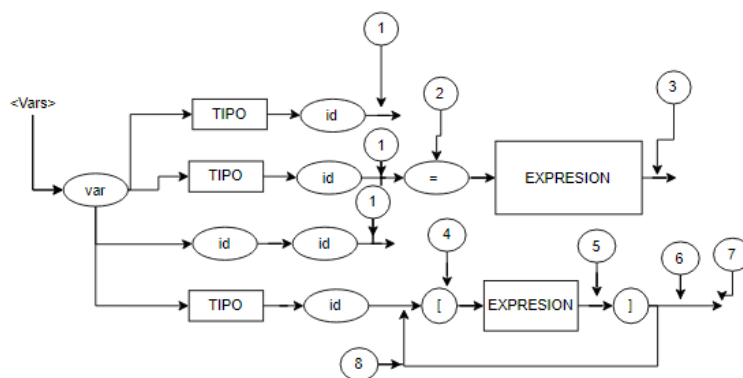
<Lectura>



```

1- np_leer:
    valor_leido = input()
    if valor_leido.type() != id.type():
        error de tipos
    else:
        quads.append([ read,,,id])
  
```

<Vars>



1- Guardar nombre y tipo en tabla correspondiente

2 - Poper.push(=)

```

3- np_asignar:
    if popper.pop == '='
        left = pilao.pop()
        right = pilao.pop()
        operador = popper.pop()
        resultType = (left, right, operador)
        if resultType != Error:
            Guardar valor de variable
            quads.append [operador, right, none, left]
    else:
        Error
  
```

4 - Variable añadir atributo de arreglo inicializado con

```

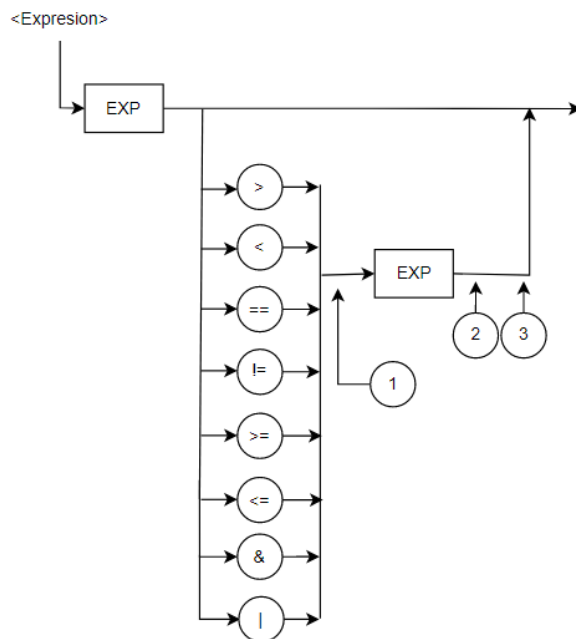
r = 1
d = 1
ls = 0
li = 0
next = none
5 - newls = pilao.pop
if newls in constTable
    addressNewLs = const(newls)
else:
    crear nueva constante de newls
    addressNewLs = const(newls)
currArr.ls = newls
actualizar R
  
```

6 -
hacer hasta llegar al ultimo nodo:
Calcular M
Calcular Size
Calcular Offset
k = offset

7 - Apartar memorias virtuales
correspondientes al size
total del arreglo

8 - Crear nueva caja y aumentar su d

<Expresion>



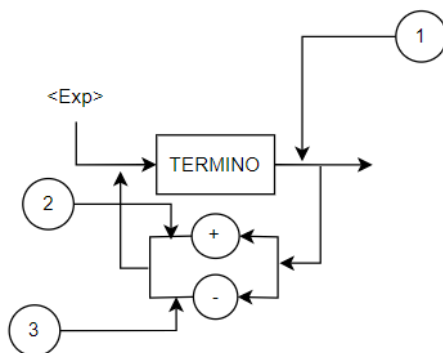
1 - Meter a poper simbolo correspondiente

2 - if Poper[-1] in ['>','<','>=','<=','!=','|']:
 right = pilaO.pop()
 left = pilaO.pop()
 operador = Poper.pop()
 resultType = getType(left,right,operador)
 if resultType != OTypeError:

currentTempMemory = apartarMemoriaTemporal(resultType)
 pilaO.append({'address':currentTempMemory, 'type':resultType})
 Quads.append([operador, left, right, currentTempMemory])
 else:
 #! Error validation
 errorType(operador, left, right)

3 - if Poper[-1] in ['|','&']:
 right = pilaO.pop()
 left = pilaO.pop()
 operador = Poper.pop()
 resultType = getType(left,right,operador)
 if resultType != OTypeError:
 currentTempMemory = apartarMemoriaTemporal(resultType)
 pilaO.append({'address':currentTempMemory, 'type':resultType})
 Quads.append([operador, left, right, currentTempMemory])
 else:
 #! Error validation
 errorType(operador, left, right)

<Exp>

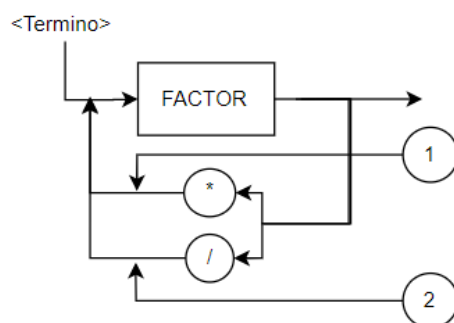


if Poper[-1] == "+" or Poper[-1] == "-":
 right = pilaO.pop()
 left = pilaO.pop()
 operador = Poper.pop()
 resultType = getType(left,right,operador)
 if resultType != OTypeError:
 currentTempMemory = apartarMemoriaTemporal(resultType)
 pilaO.append({'address':currentTempMemory, 'type':resultType})
 Quads.append([operador, left, right, currentTempMemory])
 else:
 errorType(operador, left, right)

2 - Meter el a Poper el simbolo de suma

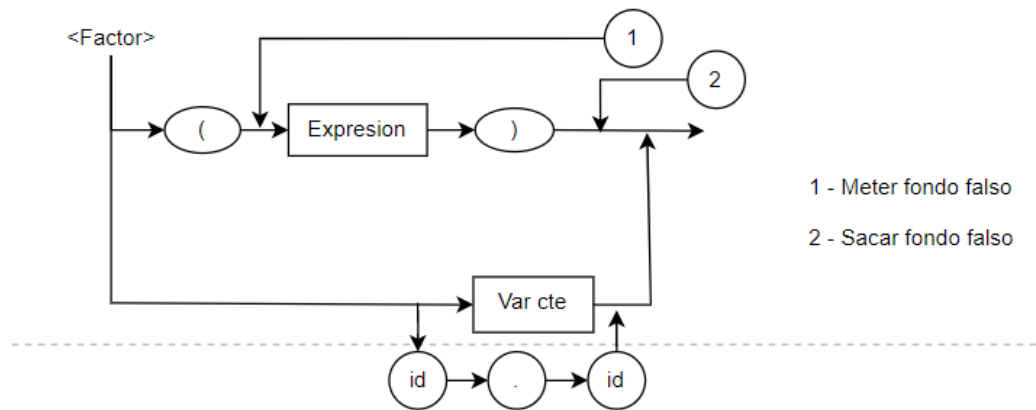
3 - Meter a Poper el simbolo de resta

<Término>

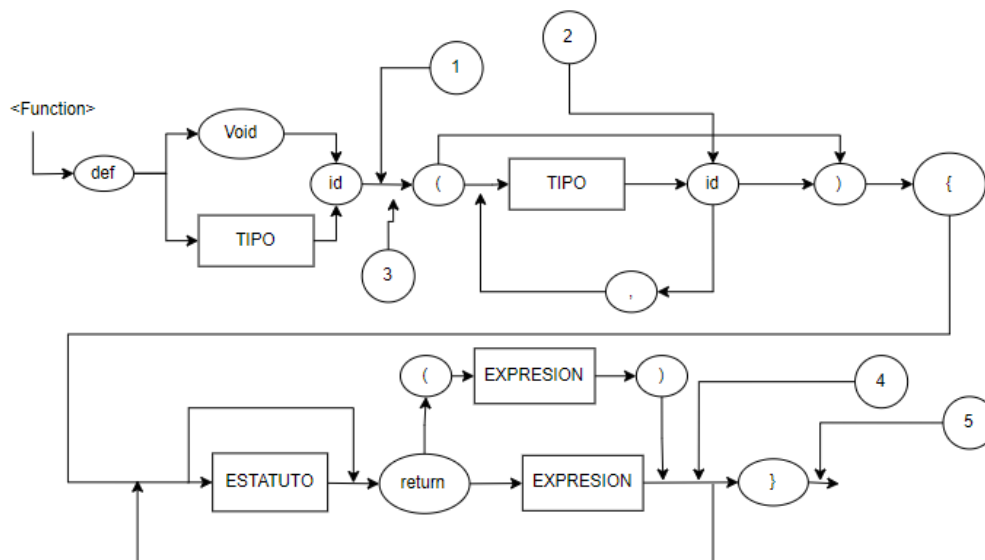


2 - Meter a Poper el simbolo de division

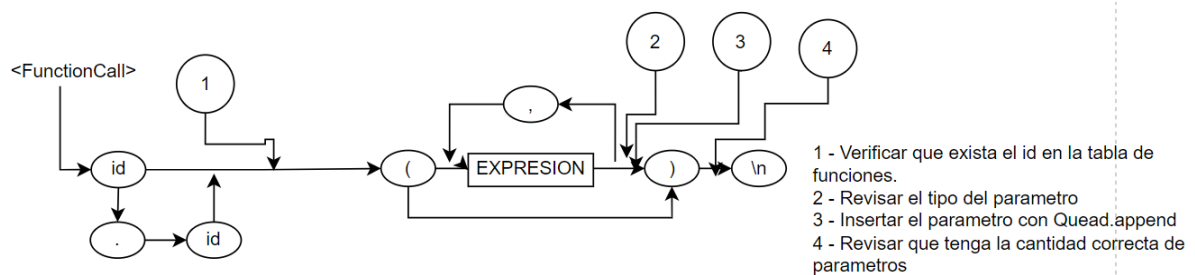
<Factor>



<Function>

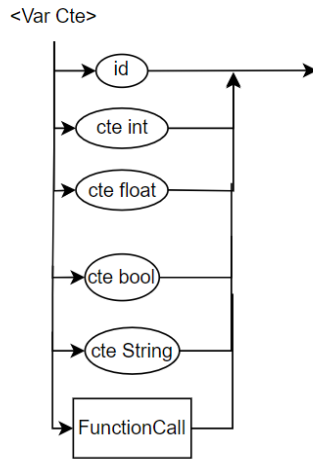


<FunctionCall>



<Var Cte>

No tiene np



- Breve descripción de cada una de las acciones semánticas y de generación de código (no más de 2 líneas).

-

Tabla de consideraciones semánticas

Ver Archivo [+ Cubosemantico](#)

O tambien cuboSemantico.py

Administración de memoria en compilación

Myglobal Vars:

Diccionario donde la llave es un string del nombre y el valor es un segundo diccionario que contiene Type, value, Scope y address

myGlobalVars	{	Var1	var2	varN	}
		{	{	{	
		type	type	type	

	value	value	value	
	scope	scope	scope	
	address	address	address	
	}	}	}	

Diccionario de funciones:

Guarda el nombre de la función así como un objeto de clase Funciones

MyDirFunctions{ Nombre : ObjetoFuncion}

Diccionario de constantes:

Para el acceso de constantes y evitar reasignación de valores que ya se encuentren declarados

myCosnta ntes	12	12000
	2.3	12001
	casa	12002
	True	12003
	46000	12004

Clase Funciones:

Contiene nombre, Línea donde inicia la función, el tipo de la función, su propio diccionario de variables locales que sigue el mismo patron de MyglobalVars, y un OrderDict para los parámetros

Este Order dict se usa para la lógica de llamado de funciones, debido a que un diccionario normal no guarda el orden de los elementos al momento de hacer una FC no se sabía la posición donde se estaban llamando, por lo que únicamente se meten en este diccionario los parámetros, siguiendo la misma lógica de los diccionarios de Variables.

myDirFunctions	name	startLinetype	varsDic	paramsDic
	"factorial"	2	{	{

			type, value, scope address	type, value, scope address
			}	}

Diccionario de objetos:

Guarda el nombre del objeto como llave y de valor es un objeto de la clase Objetos

objectDic{ nombre : Objeto de tipo "objeto" }

Clase Objetos:

Guarda nombre, diccionario de funciones y diccionario de variables

Objeto	nombre	Vars dic	FunctionsDic
		{ type, value, scope address }	{ Nombre : objeto de funcion }

PilaO:

Lista de diccionarios, donde se guarda la dirección de memoria y el tipo, evitando así la creación de pila de tipos

PilaO	Elemento 1	elemento2	
[{ address: 12200 type: int }	{ address:14000 type: String }]

Popper:

Lista de símbolos tales como: +-*/=!= etc.

[+, -, *, /, +, -, <, >]

PSaltos:

Lista de enteros

[0,12,15,20,22,41]

Cuádruplos:

Lista de listas de 4 elementos, estos elementos pueden cambiar dependiendo de la instrucción a ejecutar.

[operador, valor1, valor2, direccion]

Clase Arreglo:

Para guardar la información de un arreglo

contiene Li, Ls, rango, dimension, valor de caja, offset, nextNode, M, size , K

Arreglo	Li	Ls	Rango	D	RealSize
	Caja	Offset	Next	M	size

Máquina Virtual

Especificaciones técnicas

Equipo de computo:

Alex:

Device name LAPTOP-CJITDPM

Processor AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz

Installed RAM 8.00 GB (7.37 GB usable)

Device ID FFBFB4EE-40E7-45EF-85F6-49DAECFC9C28

Product ID 00325-82074-22926-AAOEM

System type 64-bit operating system, x64-based processor

Pen and touchNo pen or touch input is available for this display

Edition Windows 10 Home

Version 21H1

Installed on 6/2/2021

OS build 19043.1706

Serial number PF2M0539
 Experience Windows Feature Experience Pack 120.2212.4170.0
 Guasso:
 Device name DESKTOP-V73E29T
 Processor Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz
 Installed RAM 16.0 GB (15.8 GB usable)
 Device ID 46C1A479-3E21-46FF-972B-9210DE53DEB6
 Product ID 00325-95827-25195-AAOEM
 System type 64-bit operating system, x64-based processor
 Pen and touch Pen and touch support with 10 touch points

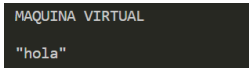
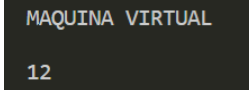
Edition Windows 10 Home
 Version 21H2
 Installed on 3/14/2021
 OS build 19044.1706
 Experience Windows Feature Experience Pack 120.2212.4170.0

Lenguaje:
 Python 3.9.5 64 bits

Librerías:
 Debido a que la lógica para ejecutar la máquina virtual depende de la generación de cuadros realizados en compilación, las librerías son las mismas. Siendo estas:

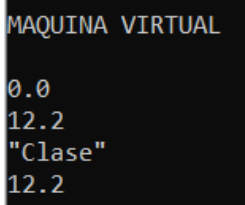
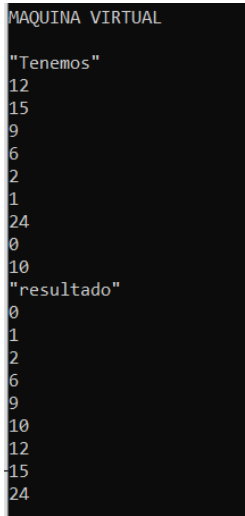
- Lark
- Collections
- copy

Pruebas de funcionalidad del lenguaje

Prueba	Archivo	Programa	Resultado esperado	Resultado real
Impresión	test_doc_impresion.txt	Start Void Main(){ Print("hola") } Finish	"Hola"	
Asignación	test_doc_asignacion.txt	Start Void Main(){ Var int num = 12 Print(num) }	12	

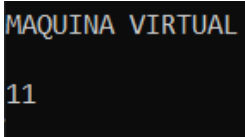
		Finish		
Condición	test_doc_condicion.txt	Start Var int aux1 = 15 Var int aux2 = 14 Void Main(){ If(aux1 > aux2){ Print("Entramos") } } Finish	"Entramos"	MAQUINA VIRTUAL "Entramos"
Lectura	test_doc_lectura.txt	Start Void Main(){ Var int rana = 1 Read(rana) Print(rana) } Finish	Input> 54	MAQUINA VIRTUAL >54 54
Ciclo	test_doc_ciclo.txt	Start Void Main(){ Var float perro = 5.0 While(perro > 0){ Print(perro) perro = perro - 1.0 } } Finish	5.0 4.0 3.0 2.0 1.0	MAQUINA VIRTUAL 5.0 4.0 3.0 2.0 1.0
Función	test_doc_funcion.txt	Start Def Void sumDos(int uno, int dos){ Print((uno + dos)) } Void Main(){ sumDos(1 , 2) } Finish		MAQUINA VIRTUAL 3
Factorial iterativo	test_fact_iter.txt	Start Def int fact(int num){ Var int result = 1 Var int index = 1 Var int limit = num + 1 While (index < limit){ result = result * index index = index + 1 } Return (result) }	24	MAQUINA VIRTUAL 24

		<pre> Void Main () { Print(fact(4)) } Finish </pre>		
Factorial recursivo	test_fact_rec ur.txt	<pre> Start Def int fact(int num){ If (num == 1){ Return (num) } Else{ Return (num * (fact(num - 1))) } } Void Main(){ Print(fact(4)) } Finish </pre>	24	<div>MAQUINA VIRTUAL</div> <div>24</div>
Find array	test_find.txt	<pre> Start Void Main(){ Var int arr[10] Print(arr:size) Var int num = arr:size /* Asignar arreglo While (num > 0) { arr[num-1] = num num = num - 1 } /* ver arreglo num = arr:size While (num > 0) { Print(arr[num-1]) num = num - 1 } /* buscar num = arr:size Var int target = 15 Var bool found = False While (num > 0) { If (arr[num-1] == target){ found = True } num = num - 1 } </pre>	11 11 10 9 8 7 6 5 4 3 2 1 False False	<div>MAQUINA VIRTUAL</div> <div> 11 11 10 9 8 7 6 5 4 3 2 1 False </div>

		Print(found) } Finish		
Arreglos en objetos y sus funciones	test_arreglo_obj_operation.txt	Start Var float myGlobarr[1] <Class> pato{ Def Void sumArrsClass(){ Var float ans = 0.0 myGlobarr[0] = 5.6 Print("Clase") Print(myGlobarr[0] + 6.6) } } Def Void sumArrs(){ Var float ans = 0.0 myGlobarr[0] = 5.6 Print(ans) Print(myGlobarr[0] + 6.6) } Void Main () { sumArrs() Var pato donald donald.sumArrsClass() } Finish	0.0 12.2 "Clase" 12.2	
Sort	test_sort.txt	Start Void Main(){ Var int arr[8] arr[0] = 12 arr[1] = 15 arr[2] = 9 arr[3] = 6 arr[4] = 2 arr[5] = 1 arr[6] = 24 arr[7] = 0 arr[8] = 10 Var int num = 0 Print("Tenemos") While (num < arr:size) { Print(arr[num]) num = num + 1	"Tenemos " 12 15 9 6 2 1 24 0 10 "resultado" 0 1 2 6 10 "resultado" " 0 1 2 6 10 12 15 24	

		<pre> } # sort Var int index = 0 While(index < arr:size){ Var int jin = 0 While (jin < (arr:size - index - 1)){ If (arr[jin] > arr[jin + 1]){ Var int aux = arr[jin+1] arr[jin + 1] = arr[jin] arr[jin] = aux } jin = jin +1 } index = index + 1 } Print("resultado") num = 0 While (num < arr:size) { Print(arr[num]) num = num + 1 } } Finish </pre>	<pre> 10 12 15 24 </pre>	
--	--	--	--------------------------	--

Factorial con arreglos	test_arreglo_fact.txt	<pre> Start Def int fact(int num){ If (num == 1){ Return (num) } Else{ Return (num * (fact(num - 1))) } } Void Main(){ Var int arr[10] Print(arr:size) Var int num = arr:size While (num > 0) { arr[num-1] = num num = num - 1 } num = arr:size While (num > 0) { Print(arr[num-1]) num = num - 1 } num = arr:size While (num > 0) { Print(fact(arr[num-1])) num = num - 1 } } Finish </pre>	<pre> 11 11 10 9 8 7 6 5 4 3 2 1 39916800 3628800 362880 40320 5040 720 120 24 6 2 1 </pre>	<p>MAQUINA VIRTUAL</p> <pre> 11 11 10 9 8 7 6 5 4 3 2 1 39916800 3628800 362880 40320 5040 720 120 24 120 24 6 2 1 </pre>
Fibonacci iterativo	test_fib_uno.txt	<pre> Start Var float punto = 5.0 Var float vaca = 6.0 Void Main(){ Var int num_deseado = 12 Var float aux = 0.0 Var float fib = 0.0 Var float fib2 = 1.0 If (num_deseado == 0){ Print(fib) } Else{ If (num_deseado == 1){ Print(fib2) </pre>	<pre> 89.0 </pre>	<p>MAQUINA VIRTUAL</p> <pre> 89.0 </pre>

		<pre> } Else{ While (num_deseado - 2 > 0) { aux = fib + fib2 fib = fib2 fib2 = aux num_deseado = num_deseado - 1 } Print(fib2) } } } Finish </pre>		
Fibonacci recursivo	test_fib_rec.t xt	<pre> Start Def int fib(int num){ If (num < 0){ Print("no se puede") } Else{ If (num == 0){ Return 0 } If ((num == 1) (num == 2)){ Return (1) } Else{ Return ((fib(num - 1)) + (fib (num - 2))) } } } Void Main(){ Print((fib(12))) } Finish </pre>	83	

Para ver los archivos ver la carpeta /Tests dentro de el repositorio de Omedetou

Documentación del código del proyecto

Puede ver las principales funciones en el documento funcionesOmedetou.py
Como mencionamos anteriormente Se programó con la extensión de pretty comments por lo que los elementos solicitados por los avances semanales están debajo de tags '#&' ejemplo:

```
#& Direction Functions
```

Y las funciones específicas de elementos se encuentran bajo los comentarios que siguen el siguiente patrón

```
'''
Inicio de puntos neurálgicos de Main
goto y relleno
'''
```

Especificaciones de comentarios:

Los comentarios siguen diferentes patrones :

#*	Comentario relevante para leer
#^	Título de comentario - especificaciones propias de lenguaje
#!	Validación de error
''' '''	Funciones necesarias para un punto específico
#?	Posibles implementaciones que no resultaron ser necesarias pero no deseamos borrar

Si bien todo el código tiene comentarios pertinentes a continuación se muestran 2 de las principales o más relevantes funciones del lenguaje Omedetou

identificador:

```
def identificador(self,tree):
    if currentObject == None:
        #* revisar que esté en local vars
        try:
            pilaO.append({
```

```

        'address':
myDirFunctions[currentFunction].varsDic[tree.children[0].value]['address'],

        'type'
:myDirFunctions[currentFunction].varsDic[tree.children[0].value]['type']

    ))

except KeyError:
    #* revisar que este en params de funcion
    try:
        pilaO.append({
            'address':
myDirFunctions[currentFunction].paramsDic[tree.children[0].value]['address'],

            'type'
:myDirFunctions[currentFunction].paramsDic[tree.children[0].value]['type']

        })

    except KeyError:
        #* Revisar si esta en global
        try:
            pilaO.append({
                'address':
myGlobalVars[tree.children[0].value]['address'],

                'type' :
myGlobalVars[tree.children[0].value]['type']

            })

        except KeyError:
            #! Error validation
            errorValueDontExist(tree)

    else:
        try:
            #* buscar en local de la funcion objetos
            pilaO.append({
                'address' :
myObjects[currentObject].funciones[currentFunction].varsDic[tree.children[0].value]['address'],

                'type':
myObjects[currentObject].funciones[currentFunction].varsDic[tree.children[0].value]['type']

            })

        except KeyError:
            #* Buscar en parametros de funcion de objeto

```

```

        try:
            pilaO.append({
                'address' :
myObjects[currentObject].funciones[currentFunction].paramsDic[tree.chil
dren[0].value]['address'],
                'type':
myObjects[currentObject].funciones[currentFunction].paramsDic[tree.chil
dren[0].value]['type']
            })
        except KeyError:
            try:
                /* Buscar en locales de objeto
                pilaO.append({
                    'address':
myObjects[currentObject].objectVarsDic[tree.children[0].value]['address
'],
                    'type'
:myObjects[currentObject].objectVarsDic[tree.children[0].value]['type']
                })
            except KeyError:
                /* Revisar si esta en global
                try:
                    pilaO.append({
                        'address':
myGlobalVars[tree.children[0].value]['address'],
                        'type' :
myGlobalVars[tree.children[0].value]['type']
                    })
                except KeyError:
                    /* Error validation
                    errorValueDontExist(tree)

```

Asignar Valor:

```

def np_asignar_valor(self,tree):
    if Poper:
        if Poper[-1] == '=':
            right = pilaO.pop()
            left = pilaO.pop()
            operador = Poper.pop()
            resultType = getType(left,right,operador)
            if currentObject == None:
                if resultType != OTypeError:

```

```

        /* Revisar en global
        if currentFunction == None:
            /* ^ GLOBAL
            /* Apartar nuevo de memoria por valor nuevo
            if myGlobalVars[left['address']]['address']
== 'tbd':

                global currentMemory
                currentMemory =
apartarMemoria(resultType)

                /* Meter el cuadrupo de la asignacion
                Quads.append([operador, right, None,
currentMemory])

                /* Registrar el valor en myGlobalVars,
no es necesario revisar que exista la llave
                /* Asignar el valor
                myGlobalVars[left['address']]['value']
= right['address']

myGlobalVars[left['address']]['address'] = currentMemory
            else:
                /* reasignar
                myGlobalVars[left['address']]['value']
= right['address']

            else:
                /* ^ LOCAL
                try: /* Asignar valores propios
                    /* Funcion a la que pertenece
                    if
myDirFunctions[currentFunction].varsDic[left['address']]['address'] ==
'tbd':

                        currentMemory =
apartarMemoriaTemporal(resultType)

                        Quads.append([operador, right,
None, currentMemory])

myDirFunctions[currentFunction].varsDic[left['address']]['value'] =
right['address']

myDirFunctions[currentFunction].varsDic[left['address']]['address'] =
currentMemory

                    else:
                        /* reasignar valor local en local

```

```

myDirFunctions[currentFunction].varsDic[left['address']]['value'] =
right['address']

                                Quads.append([operator, right, None,
myDirFunctions[currentFunction].varsDic[left['address']]['address']])
                                except KeyError:
                                    try:
                                        /* reasignar parametro value
                                        if
myDirFunctions[currentFunction].paramsDic[left['address']]['address']
== 'tbd':

                                currentMemory =
apartarMemoriaTemporal(resultType)

                                Quads.append([operator, right,
None, currentMemory])

myDirFunctions[currentFunction].paramsDic[left['address']]['value'] =
right['address']

myDirFunctions[currentFunction].paramsDic[left['address']]['address'] =
currentMemory

                                else:
                                    /* reasignar valor local en
local

myDirFunctions[currentFunction].paramsDic[left['address']]['value'] =
right['address']

                                Quads.append([operator,
right, None,
myDirFunctions[currentFunction].paramsDic[left['address']]['address']])
                                except KeyError:
                                    /* reasignar valores pero de algo
global en local

                                if
myGlobalVars[left['address']]['address'] != 'tbd':
                                Quads.append([operator,
right, None, myGlobalVars[left['address']]['address']])

myGlobalVars[left['address']]['value'] = right['address']
                                else:
                                    currentMemory =
apartarMemoria(resultType)

```



```

myGlobalVars[left['address']]['address'] = currentMemory
                                Quads.append([operator,
right, None, myGlobalVars[left['address']]['address']])

myGlobalVars[left['address']]['value'] = right['address']

        else:
            #! Error validaiton
            errorType(operator, left, right)
    else:
        #^ Buscar en objetos
        if resultType != OTypeError:
            if currentFunction == None:
                #* Crear memoria nueva
                if
myObjects[currentObject].objectVarsDic[left['address']]['address'] ==
'tbd':

myObjects[currentObject].objectVarsDic[left['address']]['address'] =
right['address']

                                currentMemory =
apartarMemoriaLocal(resultType)
                                #* Meter el cuadrupo de la asignacion
                                Quads.append([operator, right, None,
currentMemory])

                                #* Registrar el valor en myGlobalVars,
no es necesario revisar que exista la llave
                                #* Asignar el valor

myObjects[currentObject].objectVarsDic[left['address']]['value'] =
right['address']

myObjects[currentObject].objectVarsDic[left['address']]['address'] =
currentMemory

        else:
            #* reasignar

myObjects[currentObject].objectVarsDic[left['address']]['value'] =
right['address']

        else:
            #* crear memoria para una variable de una
funcion en objeto

```

```

        try: #* Asignar valores propios
            #*Funcion a la que pertenece
            if
myObjects[currentObject].funciones[currentFunction].varsDic[left['address']]
['address'] == 'tbd':
                currentMemory =
apartarMemoriaLocal(resultType)
                Quads.append([operador, right,
None, currentMemory])

myObjects[currentObject].funciones[currentFunction].varsDic[left['address']]
['value'] = right['address']

myObjects[currentObject].funciones[currentFunction].varsDic[left['address']]
['address'] = currentMemory
            else:
                #* reasignar valor local en
local

myObjects[currentObject].funciones[currentFunction].varsDic[left['address']]
['value'] = right['address']
                Quads.append([operador, right,
None,
myObjects[currentObject].funciones[currentFunction].varsDic[left['address']]
['address']])
        except KeyError:
            #* reasignar valores pero de algo
global en local de objeto
            if
myGlobalVars[left['address']]['address'] != 'tbd':
                Quads.append([operador, right, None,
myGlobalVars[left['address']]['address']])

myGlobalVars[left['address']]['value'] = right['address']
            else:
                currentMemory =
apartarMemoria(resultType)

myGlobalVars[left['address']]['address'] = currentMemory
                Quads.append([operador, right, None,
myGlobalVars[left['address']]['address']])

myGlobalVars[left['address']]['value'] = right['address']

```

Manual de usuario

Guía Rápida

Declaración

```
Var int x
```

Para la declaración, escribimos primero la palabra reservada 'Var', para definir que lo siguiente será una variable. Escribimos su tipo (que puede ser 'int', 'float', 'bool', 'string') en minúsculas, y el nombre que tendrá el identificador, el cual empezará con una letra minúscula, seguido de otras minúsculas, mayúsculas, o números.

```
Var int arr[3]
```

Si queremos hacer la declaración de un arreglo, después del identificador abrimos un bracket '[', seguido de un número entero que nos indicará el tamaño del arreglo, y cerramos con otro bracket ']'.

Asignación

```
Var float punto = 5.0
```

Para asignar un valor a una variable podemos poner seguido a su declaración el signo de igual '=', seguido del término, expresión o tipo de dato a la que queremos igualarlo.

```
aux = num1 + num2
```

Si queremos realizar una asignación después de haber declarado la variable, podemos realizarla escribiendo el identificador de la variable antes del símbolo de igual '='.

Condición

```
If (num == 1){  
    ...  
}
```

Para declarar una condición, comenzamos con la palabra reservada 'If', seguida de un par de paréntesis '()'. Entre los paréntesis pondremos una expresión o término de tipo bool. Seguido de los paréntesis, abriremos un par de llaves entre las cuales se pondrán las instrucciones que se ejecutarán en caso de que la condición se cumpla.

```
If (var){  
    ...  
}  
Else{  
    ...  
}
```

En caso de que queramos ejecutar instrucciones cuando no se cumpla la condición establecida, pondremos después de las llaves la palabra reservada 'Else', seguido de un nuevo par de llaves. Entre estas llaves colocaremos las nuevas instrucciones.

Impresión

```
Print("hola")
```

Para la impresión de un dato en terminal, se utilizará la palabra reservada 'Print', seguido de paréntesis '()'. Entre los paréntesis se pondrá el término, variable o la expresión que se desea imprimir.

Read

```
Var int rana = 1  
Read(rana)
```

Para poder asignar un input de preferencia a una variable declarada, utilizamos la palabra reservada 'Read', seguido de un par de paréntesis '()'. Entre los paréntesis escribimos el identificador de la variable a la cual le queremos asignar el valor.

Ciclo

```
Var int temp = 0  
While( temp < 10) {  
    ...  
    temp = temp + 1  
}
```

Para hacer un ciclo, empezamos poniendo la palabra reservada 'While', seguida de paréntesis '()'. Entre los paréntesis escribimos un término o expresión de tipo bool, que será la condición para entrar al ciclo. Después de los paréntesis, escribimos un par de llaves '{}', entre las cuales estarán las instrucciones que se repetirán mientras la condición del ciclo se

siga cumpliendo. Es importante que también se incluya dentro de las llaves la actualización de la condición, para que el ciclo termine en algún momento.

Función

```
Def int sumarNumeros( int a, int b){  
    ...  
    Return( ... )  
}
```

Para la declaración de una función, primero escribimos la palabra reservada 'Def', seguido de su tipo y su identificador. Después del identificador escribimos un par de paréntesis '()', los cuales contendrán los parámetros que la función utilizará. Estos parámetros contendrán su tipo de dato, y el identificador que utilizarán dentro de la función actual. Los parámetros están divididos por comas. En caso de que no utilice parámetros, no se escribirá nada entre los paréntesis.

Después de los paréntesis, se abrirán un par de llaves '{}' entre las cuales se escribirá el cuerpo de la función. Es importante recordar que en caso de no ser una función de tipo void, se tendrá que terminar el cuerpo de la función con el retorno de un dato. Este dato se retornará con la palabra reservada 'Return', seguido de un par de paréntesis '()'. Dentro de los paréntesis se podrá escribir un término o expresión, el cual deberá ser, o dar como resultado, un dato del mismo tipo que la función en la que se encuentra.

```
Def Void sumArrsClass(){  
    ...  
}
```

La función tipo void utilizará la palabra reservada 'Void' para su declaración, y no tendrá que terminar su cuerpo con el retorno de un dato.

Llamada de función

```
Def Void sumDos( int uno, int dos){  
    Print ( uno + dos )  
}  
  
Main(){  
    sumDos( 1*8+7 , 2 )  
}
```

Una vez definida la función, para poder llamarla escribimos su identificador seguido de paréntesis '()'. Dentro de los paréntesis, escribimos la cantidad de parámetros correspondientes a la expresión, separados por comas. Estos parámetros deberán ser del mismo tipo que se definió en la declaración de la función.

Comentario

```
#Asignamos un valor a la variable  
#myvar = 8
```

Para poner un comentario dentro del código, simplemente ponemos un símbolo de hashtag '#' al inicio de la línea. Todo lo que se encuentre en esa línea de código será ignorado en la compilación del programa.

Declarar Clase

```
<Class> pato {  
    Var int cuack = 10  
    Var int meme = 45  
    Var String nombre = "vacas"  
}  
Void Main(){  
    Var pato plumas  
    Print(plumas.cuack)  
}
```

Para la declaración de un objeto, escribimos la palabra reservada '<Class>' seguido de su identificador. Después ponemos un par de llaves '{}', entre las cuales escribiremos los atributos que tendrá nuestro objeto. Los atributos son declarados, y también se les puede asignar un valor.

Para la declaración del objeto, se utiliza como tipo el identificador de la clase. Para llamar a su atributo colocamos el identificador de la variable creada seguido de un punto.

Video: Manual de usuario

- https://www.youtube.com/watch?v=f751ot4GV_U