



Tecnológico de Monterrey

Diseño de Compiladores
Documentación de Lenguaje: Aang

Jimena Lomeli

Ana Jimena Lomeli Cantu
A00818665

Jorge Andrés Sabella Elizondo
A01282292

Monterrey N.L. Junio 3, 2020

Índice

Descripción del Proyecto	2
Descripción del Proyecto	2
Propósito y Objetivos	2
Alcance del Proyecto.	2
Análisis de Requerimientos y Casos de Uso generales.	2
Descripción de los principales Test Cases.	2
Bitácora del Proyecto	3
Descripción del lenguaje	8
Nombre del lenguaje:	8
Características del Lenguaje	8
Manejo de Errores	9
Descripción del compilador	10
Utilerías Especiales	10
Descripción del Análisis de Léxico y Gramatical	10
Administración de Memoria	18
Descripción de la Máquina Virtual	19
Utilerías Especiales	19
Administración de Memoria	19
Pruebas del funcionamiento del lenguaje	20
Fibonacci Recursivo	20
Fibonacci No Recursivo	21
Factorial No Recursivo	22
Factorial Recursivo	22
Sort	23
Find	25
Módulos	25
Listener.py	25
VirtualMachine.py	27
SemanticCube.py	32
Function.py	33
Variable.py	34
memory.py	35

Descripción del Proyecto

1. Descripción del Proyecto

a. Propósito y Objetivos

Nuestra visión es hacer un lenguaje de programación que sea sencillo de usar para todas las personas que quieran aprender a programar. Nuestro lenguaje de programación Aang está inspirado en la historia de Avatar: La leyenda de Aang, esto con el objetivo de motivar y atraer a los estudiantes que usarían el lenguaje desarrollado en el curso de Diseño de Compiladores.

Esto pensando que no tienen conocimientos previos sobre programación.

El objetivo de este proyecto es desarrollar un compilador de un lenguaje imperativo el cual cuenta con la sintaxis necesaria para enseñar los conceptos básicos de la programación, así como una representación visual del mismo.

b. Alcance del Proyecto.

Con el lenguaje de programación Aang los usuarios pueden crear dibujos siguiendo el formato de una cuadrícula. En el navegador el usuario puede ingresar el código, compilarlo y ejecutarlo, con esto se puede observar el dibujo creado a partir del código.

c. Análisis de Requerimientos y Casos de Uso generales.

- Con nuestro proyecto, el desarrollador puede:
- Compilar código en Aang
- Ejecutar código de Aang compilado
- Interactuar con el código de ejecución de la máquina virtual a través de una interfaz gráfica

d. Descripción de los principales Test Cases.



- i. Se diseñaron los siguientes casos para comprobar la funcionalidad del lenguaje:







Caso de Prueba	Descripción	Propósito
Fibonacci	Este programa calcula el número N de Fibonacci usando un método iterativo.	El propósito de esta prueba es comprobar que los condicionales, ciclos y asignaciones funcionan correctamente.
Fibonacci Recursivo	Este programa calcula el número N de Fibonacci usando un método recursivo.	El propósito de esta prueba es comprobar que la recursividad y condicionales funcionan correctamente. Esta prueba




		se asegura de que la máquina virtual utiliza el stack de manera adecuada.
Factorial	Este programa calcula el número N factorial (N!) usando un método iterativo.	El propósito de esta prueba es comprobar que los condicionales, ciclos y asignaciones funcionan correctamente.
Factorial Recursivo	Este programa calcula el número N factorial (N!) usando un método recursivo.	El propósito de esta prueba es comprobar la recursividad y condicionales funcionan correctamente. Esta prueba se asegura de que la máquina virtual utiliza el stack de manera adecuada.
Sort	Este programa ordena los elementos de un arreglo utilizando un algoritmo sort.	El propósito de esta prueba es comprobar que se pueden acceder elementos de un arreglo y modificarlos.
Find	Este programa encuentra el elemento X en un arreglo.	El propósito de esta prueba es comprobar el funcionamiento de los elementos estructurados.








e. Bitácora del Proyecto









i. Bitácora del Proceso

Fecha	Avance	GitHub (master branch)
Abril 24, 2020	<p>Avance #1 - Análisis Léxico y de Sintaxis</p> <p>Avance # 2 - Tabla de variables y directorio de funciones</p>	<p>Vars Table and Function Directory</p>  <p>JimenaLomeli committed on Apr 24</p>
Mayo 4, 2020	Avance # 3 - Cubo semántico	<p>Semantic Cube</p>  <p>JimenaLomeli committed 29 days ago</p>

Mayo 13, 2020	Avance #3 - Generación de Cuadрупlos de expresiones aritméticas	Added basic implementation for the arithmetic quadruple  JorgeSabella committed 21 days ago
Mayo 17, 2020	Avance # 3 - Generación de cuadрупlos para estatutos secuenciales	Added Print and assignment quadruples and fixed a bug in the Grammar  JorgeSabella committed 16 days ago
Mayo 18, 2020	Avance # 3 - Generación de cuadрупlos para estatutos secuenciales	Added Checks of semantic cube quadruple generation  JorgeSabella committed 15 days ago
Mayo 19, 2020	Avance # 4 - Generación de cuadрупlos para estatutos condicionales	Added Quadruples generation for arithmetic and if statements  JorgeSabella committed 15 days ago
Mayo 20, 2020	Avance # 4 - Generación de cuadрупlos para estatutos condicionales (ciclos)	Added memory management and while statement  JimenaLomeli committed 14 days ago
Mayo 21, 2020	Avance #5 - Generación de código de funciones	Added First part of functions statements  JorgeSabella committed 13 days ago
Mayo 23, 2020	Avance #5 - Generación de	Fixed bugs in Aang Main and

	código de funciones	in memory allocation  JorgeSabella committed 8 days ago
Mayo 27, 2020	Avance #6 - Mapa de Memoria de Ejecución para la MV	Added syntax error check  JimenaLomeli committed 6 days ago Cleaned code  JimenaLomeli committed 7 days ago Started Virtual Machine  JorgeSabella committed 7 days ago Fix bugzzz  JimenaLomeli committed 7 days ago
Mayo 28, 2020	Avance #6 - Ejecución de expresiones aritméticas	Added Virtual Machine Memory Management and arithmetic execution 

		JimenaLomeli committed 6 days ago
Mayo 29, 2020	Avance #6 - Ejecución de expresiones aritméticas y estatutos secuenciales	Added logic Operators to Virtual Machine  JorgeSabella committed 5 days ago Changed Expresion to support && and . Added Functions, conditionals...  JorgeSabella committed 5 days ago
Mayo 30, 2020	Avance #6 - Ejecución de expresiones aritméticas y estatutos secuenciales Avance #7 - Generación de código de arreglos	Added more Functions functionality to Virtual Machine  JorgeSabella committed 3 days ago
Mayo 31, 2020	Avance #7 - Generación de código de arreglos	Changed grammar to support multiple return functions, and fixed param...  JimenaLomeli committed 3 days ago Added Recursive functionality to virtual machine  JimenaLomeli committed 3 days ago Added semantics for arrays  JimenaLomeli committed 2 days ago
Junio 1, 2020	Avance #8 - Generación de código y MV para una parte de la aplicación particular	Add Django to project  JimenaLomeli committed

		<p>yesterday</p> <p>Added css to index</p>  <p>JimenaLomeli committed yesterday</p>
Junio 2, 2020	Avance #8 - Generación de código y MV para una parte de la aplicación particular	<p>Read and execute code from web page</p>  <p>JimenaLomeli committed yesterday</p> <p>Added Quadruple Generation for arrays</p>  <p>JorgeSabella committed 13 hours ago</p> <p>Added test files, find and sort</p>  <p>JorgeSabella committed 10 hours ago</p>
Junio 3, 2020	Avance #8 - Generación de código y MV para una parte de la aplicación particular	<p>Added canvas and color render</p>  <p>JimenaLomeli committed 6 hours ago</p> <p>Merge pull request #2 from JimenaLomeli/Jimena_Dev...</p>  <p>JimenaLomeli committed 6 hours ago</p> <p>Added Matrix functionality and quadruples for special functions</p>  <p>JorgeSabella committed 2 hours ago</p> <p>Added examples test cases and color quadruples</p>  <p>jimanalomeli committed 1</p>

		minute ago
--	--	------------

ii. Reflexiones

1. Jimena Lomeli

El proceso para crear un compilador fue una experiencia retadora ya que difícil entender cómo la teoría se traduce a la práctica, cómo los conceptos se traducen a código. Sin embargo, fue gratificante el proceso creativo para diseñarlo, así como ver el producto final de una manera visual. Fue divertido poder aplicar conocimientos de otras clases y juntarlos todos para realizar este proyecto.

Jimena Lomeli

Firma

2. Jorge Sabella

Este proyecto fue una increíble reto para mí. Creo que ha sido el único proyecto dentro de la universidad que me ha retado de esta manera. Esto lo digo porque para completar el proyecto fue necesario utilizar todos mis conocimientos que he adquirido durante la carrera. Esto además de que durante el proceso he aprendido demasiado y he podido identificar mis debilidades así como mis fortalezas. La verdad considero que este proyecto es la mejor manera para finalizar mis actividades estudiantiles en el Tecnológico de Monterrey.



Firma

2. Descripción del lenguaje

a. Nombre del lenguaje:

- i. Aang

b. Características del Lenguaje

El lenguaje de programación Aang cuenta con 4 funciones especiales que permiten al usuario controlar qué parte de la cuadrícula se va a colorear. Las funciones especiales son:

- Avanzar: Por default el programa empieza en el recuadro del centro, la función avanzar mueve la posición hacia la derecha.
- Pintar: Recibe como parámetro el color del cual se desea pintar el recuadro.
 - El color puede ser Azul, Rojo, Negro o Verde
- Girar: Recibe como parámetro la dirección hacia donde se quiere dirigir.
 - La dirección puede ser Arriba, Abajo, Izquierda o Derecha

c. Manejo de Errores

- i. Se obtienen errores en compilación o en ejecución si lo siguiente se realiza:

1. En compilación

- a. No se agrega un punto y coma ";" al final de cada línea de código (excepto return o cuando se declara una función).
- b. No se agrega al inicio "programa [Nombre del programa]" , seguido de "fin" al final del programa.
- c. Se utiliza una variable no declarada.
- d. Se declara una variable con el mismo nombre, declarada previamente.
- e. Se declara una función con el mismo nombre, declarada previamente.
- f. No se agrega la función principal.

2. En ejecución

- a. La dirección de memoria accesada se encuentra fuera de los rangos establecidos en el diseño.

Mensajes de Error	
Errores de Sintaxis	
Compilation ended due to syntax errors	Exception: Less Arguments Passed in to [Name] Function
Exception: More Arguments Passed in to [Name] Function	Exception: Types not match between Function call and Function parameter
Exception: [Variable Name] does not exist in the directory	Exception: [Function Name] does not exist in the directory
Exception: [Variable Name] already exists in the directory	Exception: The function already exists.
Errores de Ejecución	
Exception: Out of Range	

3. Descripción del compilador

a. Utilerías Especiales

El lenguaje de programación especificado se desarrolló utilizando como analizador léxico y sintáctico ANTLR4 con Python 3.7.2 y MacOS/Windows para el desarrollo del sistema operativo y las pruebas.

b. Descripción del Análisis de Léxico y Gramático

i. Tokens del lenguaje

Tokens	Expresión Regular
Operadores	
SUMA	'+'
RESTA	'-'
MULT	'*'
DIVISION	'/'
ASIGNAR	'='
IGUAL	'=='

DIFERENTE	'!='
MENOR	'<'
MAYOR	'>'
Y_SIMBOLO	'&&'
O_SIMBOLO	' '
KeyWords	
INT	'int'
CHAR	'char'
BOOL	'bool'
IF	'if'
ELSE	'else'
WHILE	while
PRINT	print
RETURN	return
PROGRAMA	programa
EMPEZAR	empezar
FIN	fin
Separadores	
I_PARENTHESIS	'('
D_PARENTHESIS	')'
I_CORCHETE	'{'
D_CORCHETE	'}'
PYCOMA	','
COMA	','
Tipos de datos	
VOID	void
CTE_BOOL	'True' 'False'
ID	[a-zA-Z]+

CTE_INT	[0-9][0-9]*
CTE_CHAR	" [A-Za-z] "
Espacios y comentarios	
COMENT	'# ~[\r\n]* -> skip
WHITESPACE	[\t\r\n]+ -> skip

ii. Gramática Formal

```

programa: PROGRAMA ID PYCOMA p1 FIN;
p1: variable p2
    | funcion p2
    | principal;
p2: funcion p2
    | principal;

variable: tipo_id v;

principal: tipo_id EMPEZAR I_CORCHETE
bloque D_CORCHETE;

v: ID v1 PYCOMA v2
    | ID I_LLAVE exp D_LLAVE v1 PYCOMA
v2;

v1: COMA ID v1
    | COMA ID I_LLAVE exp D_LLAVE v1
    | /* epsilon */;
v2: variable | /* epsilon */;

tipo_id: INT
        | CHAR
        | BOOL;

funcion:
    f ID I_PARENTESIS f1 D_PARENTESIS
I_CORCHETE bloque D_CORCHETE;

f: VOID
    | tipo_id;
f1: tipo_id ID f2
    | /* epsilon */;

asignacion: ID ASIGNAR a PYCOMA;

a: expresion
    | llamar_fun;

expresion: exp e
            | exp e Y_SIMBOLO
expresion
            | exp e O_SIMBOLO
expresion
e: MAYOR exp
    | MENOR exp
    | IGUAL exp
    | DIFERENTE exp
    | /* epsilon */;

exp: termino e1;
e1: SUMA termino e1
    | RESTA termino e1
    | /* epsilon */;

factor: I_PARENTESIS expresion
D_PARENTESIS
        | cte_var;

termino: factor t;
t: MULT factor t
    | DIVISION factor t
    | /* epsilon */;

condicion:
    IF I_PARENTESIS expresion
D_PARENTESIS c1;

c1: I_CORCHETE acciones D_CORCHETE
c2;

```

```

f2: COMA tipo_id ID f2
    | /* epsilon */;

bloque: variable bloque
    | acciones
    | /* epsilon */;

acciones: asignacion acciones
    | condicion acciones
    | ciclo1 acciones
    | escribir acciones
    | llamar_fun acciones
    | pintar acciones
    | mover acciones
    | cambiar acciones
    | fun_regresar acciones
    | /* epsilon */;

pintar: PINTAR I_PARENTESIS
D_PARENTESIS PYCOMA;

mover: MOVER I_PARENTESIS exp
D_PARENTESIS PYCOMA;

cambiar:
    CAMBIAR_DIRECCION I_PARENTESIS
ARRIBA D_PARENTESIS PYCOMA
    | CAMBIAR_DIRECCION I_PARENTESIS
ABAJO D_PARENTESIS PYCOMA
    | CAMBIAR_DIRECCION I_PARENTESIS
DERECHA D_PARENTESIS PYCOMA
    | CAMBIAR_DIRECCION I_PARENTESIS
IZQUIERDA D_PARENTESIS PYCOMA;

fun_regresar: RETURN exp PYCOMA
    | RETURN llamar_fun;

c2: ELSE I_CORCHETE acciones
D_CORCHETE
    | /* epsilon */;

ciclo1:
    WHILE I_PARENTESIS expresion
D_PARENTESIS ciclo2;

ciclo2: I_CORCHETE acciones
D_CORCHETE;

escribir: PRINT I_PARENTESIS es
D_PARENTESIS PYCOMA;

es: expresion es2
    | CTE_CHAR es2
    | llamar_fun;

es2: COMA es
    | /* epsilon */;

cte_var: CTE_INT
    | CTE_CHAR
    | ID
    | CTE_BOOL;

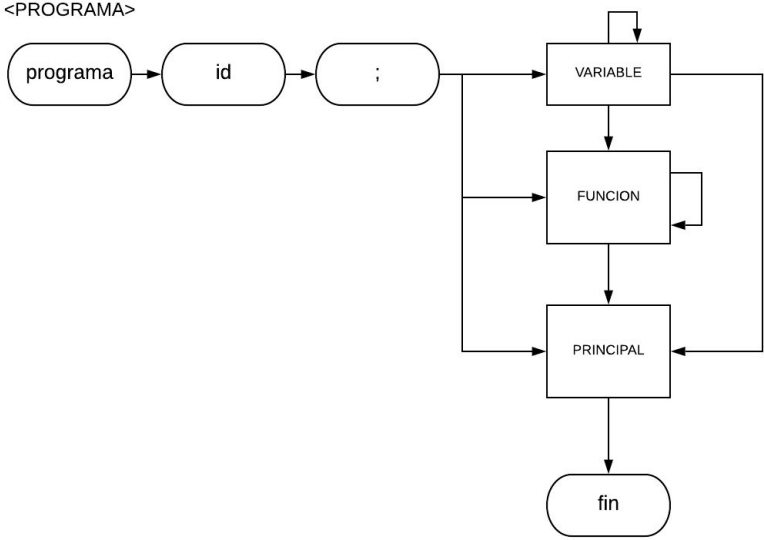
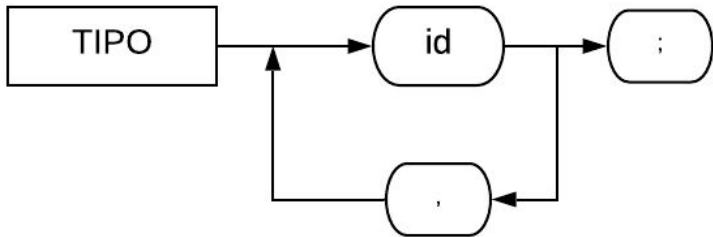
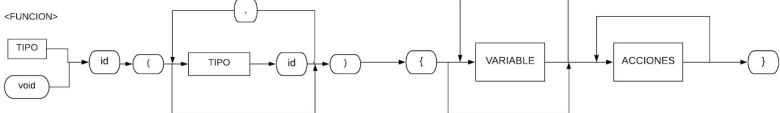
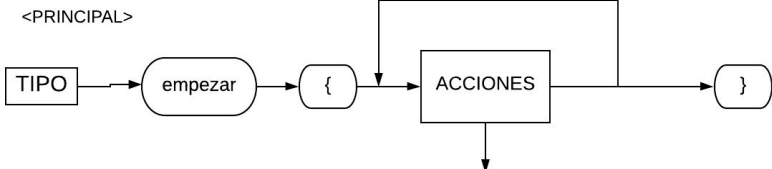
llamar_fun: ID I_PARENTESIS
argumentos D_PARENTESIS;

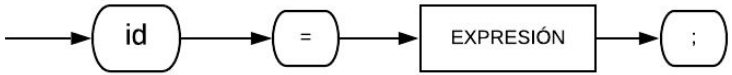
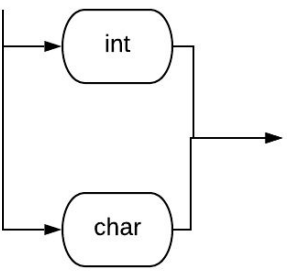
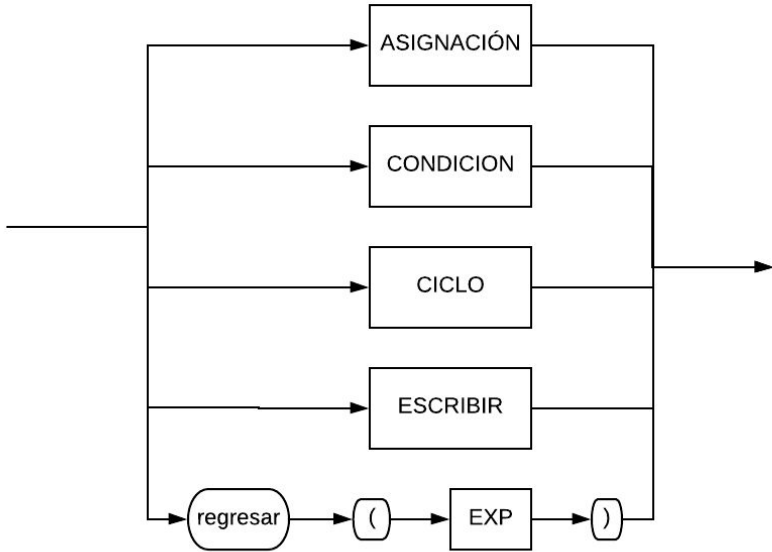
argumentos: exp agregar_args
    | /* epsilon */;

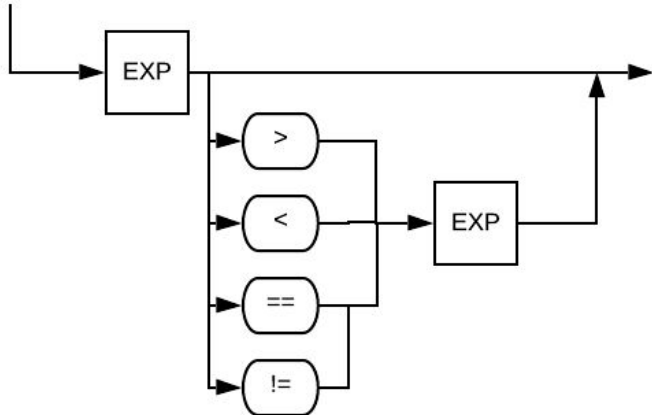
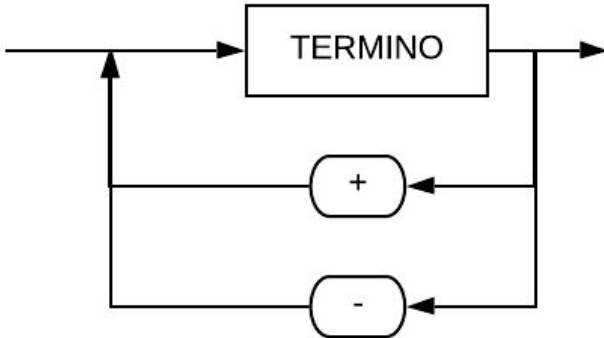
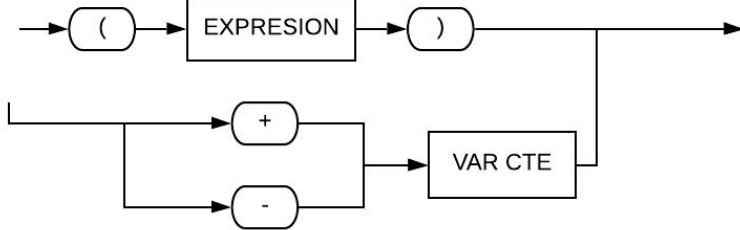
agregar_args: COMA exp agregar_args
    | /* epsilon */;

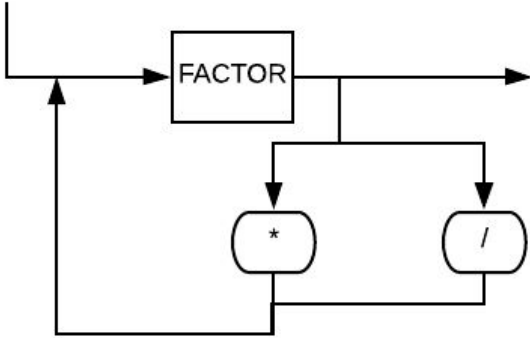
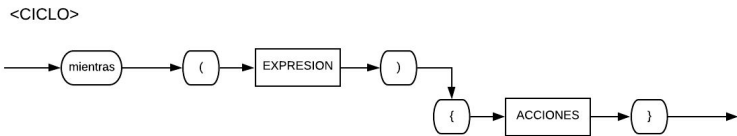
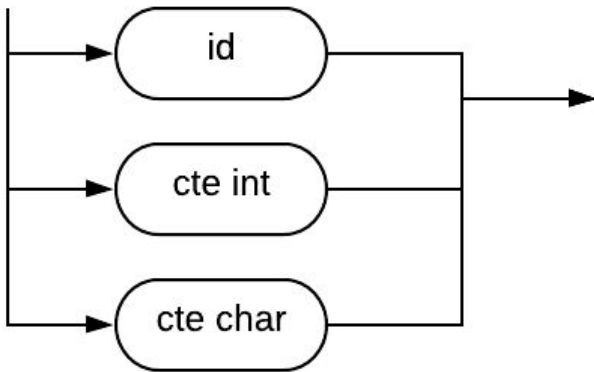
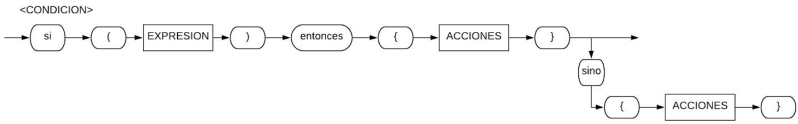
```

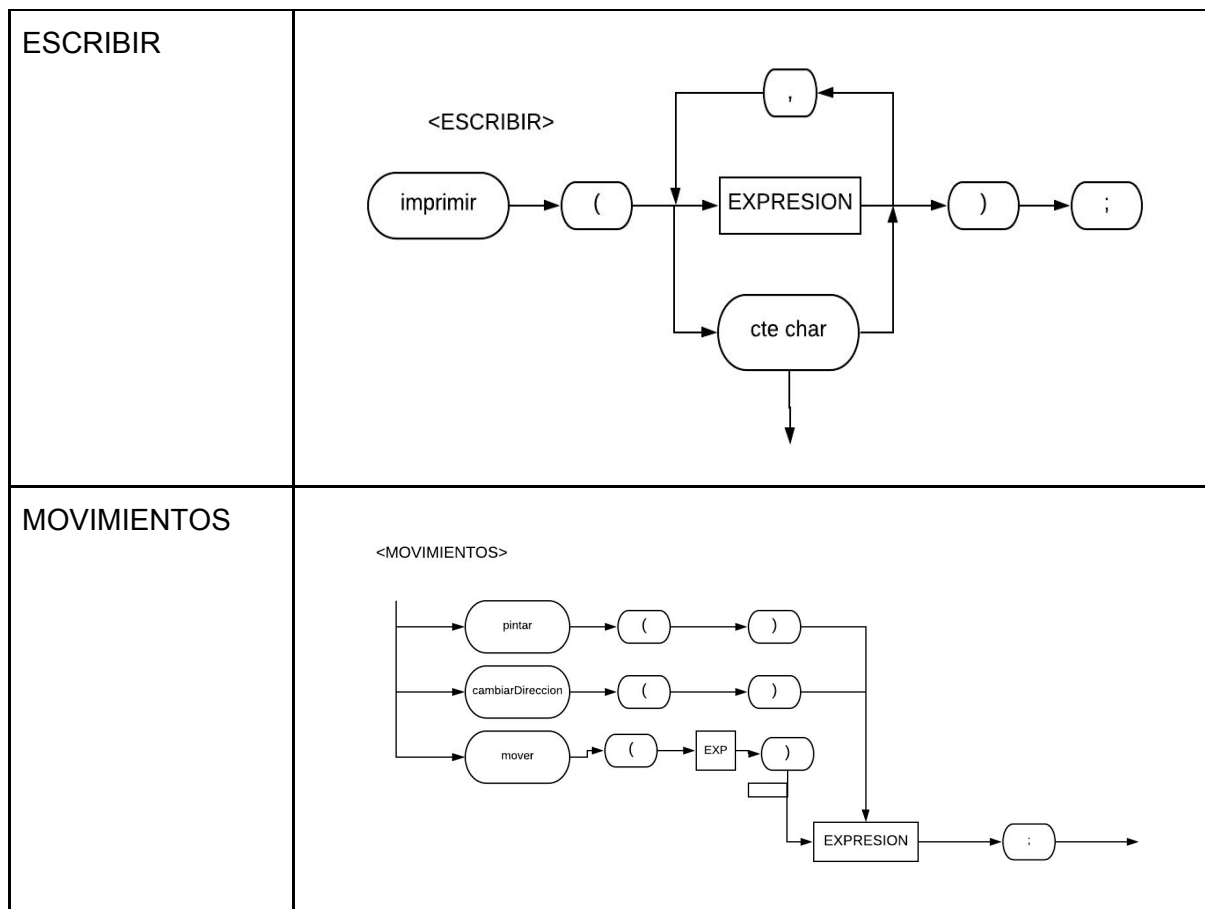
iii. Diagramas de Sintaxis

Acciones	Diagrama de Sintaxis
PROGRAMA	<p data-bbox="576 412 711 434"><PROGRAMA></p>  <pre> graph LR programa([programa]) --> id1([id]) id1 --> semicolon([;]) semicolon --> VARIABLE[VARIABLE] semicolon --> FUNCION[FUNCION] semicolon --> PRINCIPAL[PRINCIPAL] VARIABLE --> VARIABLE VARIABLE --> PRINCIPAL FUNCION --> FUNCION FUNCION --> PRINCIPAL PRINCIPAL --> fin([fin]) </pre>
VARIABLE	<p data-bbox="620 1095 815 1128"><VARIABLE></p>  <pre> graph LR TIPO1[TIPO] --> id([id]) TIPO1 --> comma([,]) id --> TIPO2[TIPO] comma --> TIPO2 TIPO2 --> semicolon([;]) </pre>
FUNCION	<p data-bbox="560 1520 616 1543"><FUNCION></p>  <pre> graph LR TIPO1[TIPO] --> id1([id]) void([void]) --> id1 id1 --> LP([(]) LP --> TIPO2[TIPO] LP --> id2([id]) TIPO2 --> comma([,]) id2 --> comma comma --> RP([)]) RP --> LBRACE([{]) RP --> VARIABLE[VARIABLE] LBRACE --> ACCIONES[ACCIONES] VARIABLE --> ACCIONES ACCIONES --> RBRACE([}]) </pre>
PRINCIPAL	<p data-bbox="576 1711 695 1733"><PRINCIPAL></p>  <pre> graph LR TIPO[TIPO] --> empezar([empezar]) empezar --> LBRACE([{]) LBRACE --> ACCIONES[ACCIONES] ACCIONES --> ACCIONES ACCIONES --> RBRACE([}]) ACCIONES --> exit(()) </pre>

ASIGNACION	<p><ASIGNACIÓN></p>  <pre> graph LR Start(()) --> id(id) id --> eq(=) eq --> EXP[EXPRESIÓN] EXP --> semicolon(;) </pre>
TIPO	<p><TIPO></p>  <pre> graph LR Start(()) --> Split(()) Split --> int(int) Split --> char(char) int --> Join(()) char --> Join Join --> End(()) </pre>
ACCIONES	<p><ACCIONES></p>  <pre> graph LR Start(()) --> Split(()) Split --> AS[ASIGNACIÓN] Split --> CON[CONDICION] Split --> CIC[CICLO] Split --> ESCR[ESCRIBIR] Split --> Reg(regresar) Reg --> LP("(") LP --> EXP[EXP] EXP --> RP(")") AS --> Join(()) CON --> Join CIC --> Join ESCR --> Join RP --> Join Join --> End(()) </pre>

EXPRESION	<p><EXPRESIÓN></p> 
EXP	<p><EXP></p> 
FACTOR	<p><FACTOR></p> 

TERMINO	<p><TERMINO></p> 
CICLO	<p><CICLO></p> 
VAR_CTE	<p><VAR CTE></p> 
CONDICION	<p><CONDICION></p> 



iv. Tabla de consideraciones semánticas.

Las siguientes operaciones aritméticas son soportadas por el lenguaje Aang (el orden de los operandos no es importante):

Operando Izquierdo	Operador	Operando Derecho	Resultado
int	+ , - , / , * , =	int	int
bool	> , < , != , == , = , && ,	bool	bool
char	== , != , =	char	char

c. Administración de Memoria

Memoria Compilación	
Estructura	Tipo
PilaO	stack()
PilaOper	stack()
Avail	stack()
PSaltos	stack()
PilaTipos	stack()
PilaFunc	stack()
PilaFuncParam	stack()
PilaArr	stack()
memoriaGlobal	memory()
memoriaConstante	memory()
memoriaTemporal	memory()

Tipo	Primera dirección	Última dirección
Global	1000	3000
Entero	1000	1999
Bool	2000	2999
Char	3000	3999
Temporal	0	0
Constante	7000	9999
Entero	7000	7999
Bool	8000	8999
Char	9000	9999
Temporal	0	0
Temporal	10000	12999
Entero	10000	10999
Bool	11000	11999
Char	12000	12999
Temporal	0	0

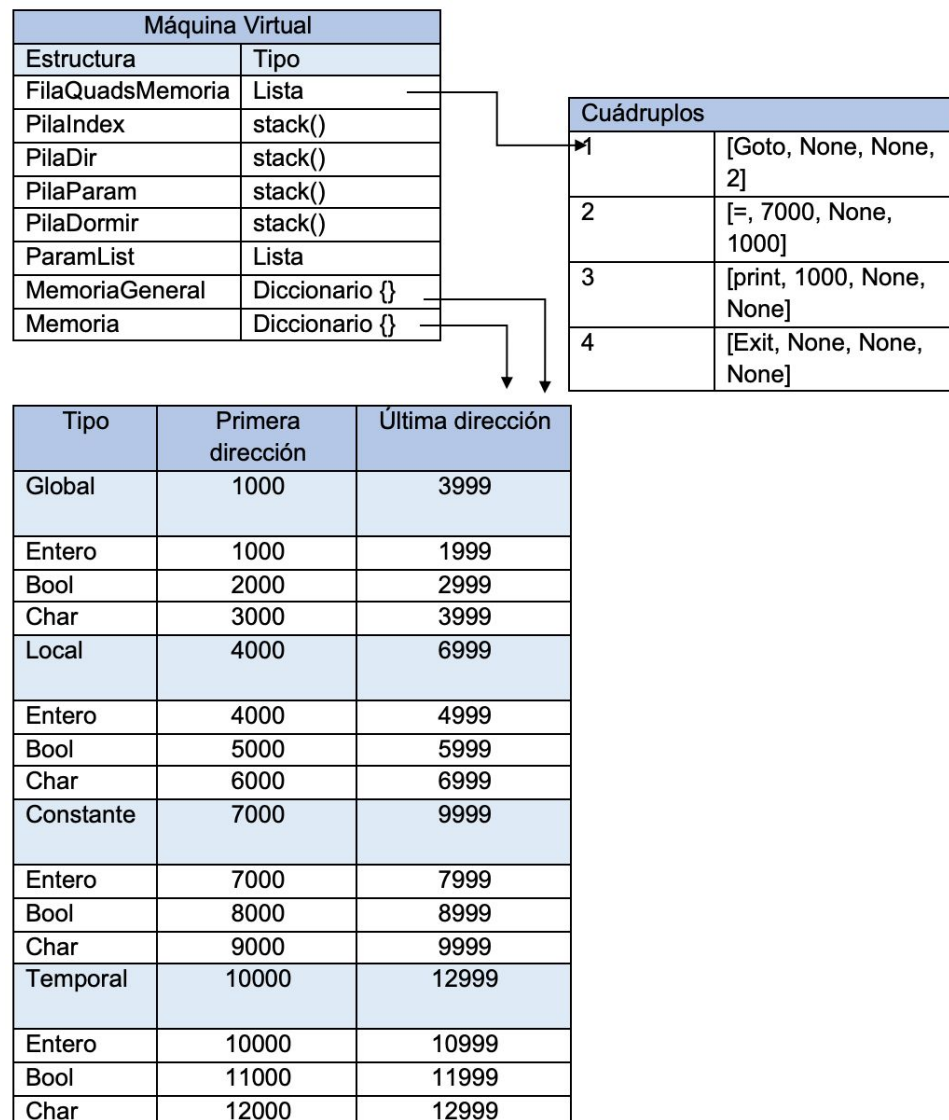
4. Descripción de la Máquina Virtual

a. Utilerías Especiales

Como herramienta de serialización utilizamos *pickle* la cual implementa protocolos binarios para serializar y deserializar una estructura de objetos en Python. De esta manera, nosotros importamos a la máquina virtual los cuádruplos generados en la compilación.

b. Administración de Memoria

El lenguaje de programación Python te permite crear diccionarios dentro de otros diccionarios, Aang aprovecha esta estructura para soportar de una manera ordenada el manejo de scopes en la memoria.



- El archivo Variable.py maneja las funciones relacionadas a las variables, una variable tiene como parte de su estructura el scope.

- El archivo memory.py maneja los contadores necesarios para saber la cantidad de variables enteras, char, booleanas o temporales hay. Asimismo, se manejan aquí las funciones relacionadas a esto.
- El archivo Function.py identifica como parte de su estructura

5. Pruebas del funcionamiento del lenguaje

a. Fibonacci Recursivo

Código	Código Intermedio	Ejecución
<pre> programa fibonacci; int x, i; int fib (int x) { int a, b, c; if(x == 1 x==0) { return(x); } else { a = fib(x-1); b = fib(x-2); c = a + b; return c; } } int empezar { x=6; i=0; while(i < x) { i=i+1; print(fib(i)); } } fin </pre>	<pre> 1 [Goto, None, None, 24] 2 [==, 4000, 7000, 11000] 3 [==, 4000, 7001, 11001] 4 [, 11000, 11001, 11002] 5 [GotoF, 11002, None, 8] 6 [RETURN, None, None, 4000] 7 [Goto, None, None, 23] 8 [ERA, None, None, fib] 9 [-, 4000, 7000, 10000] 10 [PARAMETER, 10000, None, Par1] 11 [GOSUB, fib, None, None] 12 [=*, fib, None, 10001] 13 [=, 10001, None, 4001] 14 [ERA, None, None, fib] 15 [-, 4000, 7003, 10002] 16 [PARAMETER, 10002, None, Par1] 17 [GOSUB, fib, None, None] 18 [=*, fib, None, 10003] 19 [=, 10003, None, 4002] 20 [+ , 4001, 4002, 10004] 21 [=, 10004, None, 4003] 22 [RETURN, None, None, 4003] 23 [EndFunc, None, None, None] 24 [=, 7004, None, 1000] 25 [=, 7001, None, 1001] 26 [<, 1001, 1000, 11003] 27 [GotoF, 11003, None, 36] 28 [+ , 1001, 7000, 10005] 29 [=, 10005, None, 1001] 30 [ERA, None, None, fib] 31 [PARAMETER, 1001, None, Par1] 32 [GOSUB, fib, None, None] </pre>	8

	<pre> 33 [=*, fib, None, 10006] 34 [print, 10006, None, None] 35 [Goto, None, None, 26] 36 [Exit, None, None, None] </pre>	
--	----------------------------------------------------------------------------------------------------------------------------	--

b. Fibonacci No Recursivo

Código	Código Intermedio	Ejecución
<pre> programa fibo; int num, numero, next; int n, i, res; int fib(int x){ num = 0; numero = 1; next = 1; i=0; while(i < (x - 1)){ i = i+1; next = num + numero; num = numero; numero = next; } return next; } int empezar{ n = 6; res = fib(n); print(res); } fin </pre>	<pre> 1 [Goto, None, None, 18] 2 [=, 7000, None, 1000] 3 [=, 7001, None, 1001] 4 [=, 7001, None, 1002] 5 [=, 7000, None, 1004] 6 [-, 4000, 7001, 10000] 7 [<, 1004, 10000, 11000] 8 [GotoF, 11000, None, 16] 9 [+ , 1004, 7001, 10001] 10 [=, 10001, None, 1004] 11 [+ , 1000, 1001, 10002] 12 [=, 10002, None, 1002] 13 [=, 1001, None, 1000] 14 [=, 1002, None, 1001] 15 [Goto, None, None, 6] 16 [RETURN, None, None, 1002] 17 [EndFunc, None, None, None] 18 [=, 7006, None, 1003] 19 [ERA, None, None, fib] 20 [PARAMETER, 1003, None, Par1] 21 [GOSUB, fib, None, None] 22 [=*, fib, None, 10003] 23 [=, 10003, None, 1005] 24 [print, 1005, None, None] 25 [Exit, None, None, None] </pre>	8

c. Factorial No Recursivo

Código	Código Intermedio	Ejecución
--------	-------------------	-----------

<pre> programa factNR; int n, fact, i; int empezar { n=6; i=1; fact=1; if (n<0) { print('W'); } else { while(i < (n+1)) { fact=fact*i; i = i+1; } print(fact); } } fin </pre>	<pre> 1 [Goto, None, None, 2] 2 [=, 7000, None, 1000] 3 [=, 7001, None, 1002] 4 [=, 7001, None, 1001] 5 [<, 1000, 7003, 11000] 6 [GotoF, 11000, None, 9] 7 [print, 9000, None, None] 8 [Goto, None, None, 17] 9 [+ , 1000, 7001, 10000] 10 [<, 1002, 10000, 11001] 11 [GotoF, 11001, None, 17] 12 [*, 1001, 1002, 10001] 13 [=, 10001, None, 1001] 14 [+ , 1002, 7001, 10002] 15 [=, 10002, None, 1002] 16 [Goto, None, None, 9] 17 [print, 1001, None, None] 18 [Exit, None, None, None] </pre>	720
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----

d. Factorial Recursivo

Código	Código Intermedio	Ejecución
<pre> int a; int n; int factorial(int n) { int b; if(n < 0) { return 0; } if(n > 1) { b = factorial(n-1); a = n * b; return a; } return 1; } int empezar { n=6; print(factorial(n)); } fin </pre>	<pre> 1 [Goto, None, None, 18] 2 [<, 4000, 7000, 11000] 3 [GotoF, 11000, None, 5] 4 [RETURN, None, None, 7000] 5 [>, 4000, 7002, 11001] 6 [GotoF, 11001, None, 16] 7 [ERA, None, None, factorial] 8 [-, 4000, 7002, 10000] 9 [PARAMETER, 10000, None, Par1] 10 [GOSUB, factorial, None, None] 11 [=*, factorial, None, 10001] 12 [=, 10001, None, 4001] 13 [*, 4000, 4001, 10002] 14 [=, 10002, None, 1000] 15 [RETURN, None, None, 1000] 16 [RETURN, None, None, 7002] 17 [EndFunc, None, None, None] 18 [=, 7005, None, 1001] 19 [ERA, None, None, </pre>	720

	factorial] 20 [PARAMETER, 1001, None, Parl] 21 [GOSUB, factorial, None, None] 22 [=*, factorial, None, 10003] 23 [print, 10003, None, None] 24 [Exit, None, None, None]	
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

e. Sort

Código	Código Intermedio	Ejecución
<pre> programa sortArray; int A[5], i, key, j; int empezar { A[0] = 12; A[1] = 0; A[2] = 39; A[3] = 50; A[4] = 1; i=0; while (i < 5){ key= i; j= i+1; while (j < 5){ if (A[key] > A[j]){ key=j; } j = j+ 1; } A[i]=A[key]; A[key] = A[i]; i= i+ 1; } } fin </pre>	<pre> 1 [Goto, None, None, 2] 2 [+, 1000, 7001, -10000] 3 [Ver, 7001, 0, 5] 4 [=, 7002, None, -10000] 5 [+, 1000, 7003, -10001] 6 [Ver, 7003, 0, 5] 7 [=, 7001, None, -10001] 8 [+, 1000, 7005, -10002] 9 [Ver, 7005, 0, 5] 10 [=, 7006, None, -10002] 11 [+, 1000, 7007, -10003] 12 [Ver, 7007, 0, 5] 13 [=, 7008, None, -10003] 14 [+, 1000, 7009, -10004] 15 [Ver, 7009, 0, 5] 16 [=, 7003, None, -10004] 17 [=, 7001, None, 1005] 18 [<, 1005, 7000, 11000] 19 [GotoF, 11000, None, 26] 20 [+, 1000, 1005, -10005] 21 [Ver, 1005, 0, 5] 22 [print, -10005, None, None] 23 [+ , 1005, 7003, 10006] 24 [=, 10006, None, 1005] 25 [Goto, None, None, 18] 26 [print, 9000, None, None] 27 [=, 7001, None, 1005] 28 [=, 7001, None, 1007] 29 [=, 7001, None, 1008] 30 [<, 1005, 7000, 11001] 31 [GotoF, 11001, None, 60] 32 [+ , 1005, 7003, 10007] 33 [=, 10007, None, 1007] 34 [<, 1007, 7000, 11002] 35 [GotoF, 11002, None, 57] 36 [print, 9001, None, None] </pre>	<pre> 50 39 12 1 0 </pre>

	<pre> 37 [+, 1000, 1005, -10008] 38 [Ver, 1005, 0, 5] 39 [+, 1000, 1007, -10009] 40 [Ver, 1007, 0, 5] 41 [<, -10008, -10009, 11003] 42 [GotoF, 11003, None, 54] 43 [+, 1000, 1005, -10010] 44 [Ver, 1005, 0, 5] 45 [=, -10010, None, 1008] 46 [+, 1000, 1007, -10011] 47 [Ver, 1007, 0, 5] 48 [+, 1000, 1005, -10012] 49 [Ver, 1005, 0, 5] 50 [=, -10011, None, -10012] 51 [+, 1000, 1007, -10013] 52 [Ver, 1007, 0, 5] 53 [=, 1008, None, -10013] 54 [+, 1007, 7003, 10014] 55 [=, 10014, None, 1007] 56 [Goto, None, None, 34] 57 [+, 1005, 7003, 10015] 58 [=, 10015, None, 1005] 59 [Goto, None, None, 30] 60 [=, 7001, None, 1005] 61 [<, 1005, 7000, 11004] 62 [GotoF, 11004, None, 69] 63 [+, 1000, 1005, -10016] 64 [Ver, 1005, 0, 5] 65 [print, -10016, None, None] 66 [+, 1005, 7003, 10017] 67 [=, 10017, None, 1005] 68 [Goto, None, None, 61] 69 [Exit, None, None, None] </pre>	
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

f. Find

Código	Código Intermedio	Ejecución
<pre> programa findArray; int i, A[5], num, cnt, pos; int empezar { num = 39; pos = 0; i=0; cnt = 0; A[0] = 12; A[1] = 0; A[2] = 39; A[3] = 50; A[4] = 1; </pre>	<pre> 1 [Goto, None, None, 2] 2 [=, 7001, None, 1006] 3 [=, 7002, None, 1008] 4 [=, 7002, None, 1000] 5 [=, 7002, None, 1007] 6 [+, 1001, 7002, -10000] 7 [Ver, 7002, 0, 5] 8 [=, 7006, None, -10000] 9 [+, 1001, 7007, -10001] 10 [Ver, 7007, 0, 5] 11 [=, 7002, None, -10001] 12 [+, 1001, 7009, -10002] 13 [Ver, 7009, 0, 5] 14 [=, 7001, None, -10002] </pre>	2

<pre> while (i<5) { if(A[i]==num) { cnt=1; pos=i+1; } i = i+1; } if(cnt==0) { print(False); } else { print(pos - 1); } } fin </pre>	<pre> 15 [+*, 1001, 7011, -10003] 16 [Ver, 7011, 0, 5] 17 [=, 7012, None, -10003] 18 [+*, 1001, 7013, -10004] 19 [Ver, 7013, 0, 5] 20 [=, 7007, None, -10004] 21 [<, 1000, 7000, 11000] 22 [GotoF, 11000, None, 33] 23 [+*, 1001, 1000, -10005] 24 [Ver, 1000, 0, 5] 25 [==, -10005, 1006, 11001] 26 [GotoF, 11001, None, 30] 27 [=, 7007, None, 1007] 28 [+ , 1000, 7007, 10006] 29 [=, 10006, None, 1008] 30 [+ , 1000, 7007, 10007] 31 [=, 10007, None, 1000] 32 [Goto, None, None, 21] 33 [==, 1007, 7002, 11002] 34 [GotoF, 11002, None, 37] 35 [print, 8000, None, None] 36 [Goto, None, None, 39] 37 [-, 1008, 7007, 10008] 38 [print, 10008, None, None] 39 [Exit, None, None, None] </pre>	
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

6. Módulos

a. Listener.py

En este archivo manejamos la creación de cuádruplos en la fase de compilación, se sobrescribe el AangListener.py. En este archivo utilizamos ciertas funciones que nos permiten identificar los puntos neurálgicos y así generar los cuádruplos correctamente. De igual manera se incluyen en este archivo la creación de cuádruplos para las funciones especiales.

```

class AangCustomListener(AangListener):
    PilaO = stack()
    PilaOper = stack()
    Avail = avail()
    PSaltos = stack()
    FilaQuads = []
    FilaQuadsMemoria = []
    PilaTipos = stack()
    PilaFunc = stack()
    PilaFuncParam = stack()
    PilaArr = stack()
    ParameterCounter = 0
    TempParameters = []
    memoriaGlobal = memory(1000, 2000, 3000, 0)
    memoriaConstante = memory(7000, 8000, 9000, 0)
    memoriaTemporal = memory(10000, 11000, 12000, 0)

```

```

# Memory address
# global varTable
global varIntAddr
global varCharAddr

functionDirectory = FunctionDir()
varTable = VariableTable({})
constTable = ConstantTable()
localVarTable = VariableTable({})

# ===== CUADRUPLAS PARA VM =====
pickle_out = open("Cuadruplos.pickle", "wb")
pickle.dump(self.FilaQuadsMemoria, pickle_out)
pickle.dump(self.functionDirectory, pickle_out)
pickle.dump(self.constTable, pickle_out)
pickle_out.close()

#===== FUNCIONES ESPECIALES =====

def enterPintar(self, ctx: AangParser.PintarContext):
    operator = "pintar"
    rightOperand = None
    leftOperand = None
    result = None
    quad = Quadruple(
        operator, rightOperand, leftOperand, result)
    quad2 = Quadruple(
        operator, rightOperand, leftOperand, result)
    self.FilaQuads.append(quad)
    self.FilaQuadsMemoria.append(quad2)

def exitPintar(self, ctx: AangParser.PintarContext):
    pass

def enterMover(self, ctx: AangParser.MoverContext):
    pass

def exitMover(self, ctx: AangParser.MoverContext):
    operator = "mover"
    rightOperand = self.PilaO.pop()
    leftOperand = None
    result = None
    quad = Quadruple(
        operator, rightOperand[0], leftOperand, result)
    quad2 = Quadruple(
        operator, rightOperand[2], leftOperand, result)
    self.FilaQuads.append(quad)
    self.FilaQuadsMemoria.append(quad2)
    pass

def enterCambiar(self, ctx: AangParser.CambiarContext):
    operator = "cambiar"
    leftOperand = None

```

```

result = None

if ctx.ARRIBA() != None:
    rightOperand = 'Arriba'

elif ctx.ABAJO() != None:
    rightOperand = 'Abajo'

elif ctx.DERECHA() != None:
    rightOperand = 'Derecha'

else:
    rightOperand = 'Izquierda'
quad = Quadruple(
    operator, rightOperand, leftOperand, result)
quad2 = Quadruple(
    operator, rightOperand, leftOperand, result)
self.FilaQuads.append(quad)
self.FilaQuadsMemoria.append(quad2)
pass

def exitCambiar(self, ctx: AangParser.CambiarContext):
    pass

```

b. VirtualMachine.py

Este archivo se utiliza para la fase de ejecución, es aquí donde se itera sobre cada cuádruplo, se identifican los operandos y operadores y se realiza la expresión correspondiente. En este archivo se importan los objetos de compilación y se crean para la serialización de la matriz de colores, esto utilizando la librería pickle.

```

from stack import stack

```

```

import sys
import pickle
import copy

```

```

Memoria = {
    "Global": {
        "Entero": {},
        "Char": {},
        "Bool": {}
    },
    "Local": {
        "Entero": {},
        "Char": {},
        "Bool": {}
    },
    "Constante": {
        "Entero": {},
        "Char": {},
        "Bool": {}
    },
}

```

```

    "Temporal": {
        "Entero": {},
        "Char": {},
        "Bool": {}
    }
}

# ===== IMPORTAR OBJETOS DE COMPILACION =====
pickle_in = open("Cuadruplos.pickle", "rb")
FilaQuadsMemoria = pickle.load(pickle_in)
functionDirectory = pickle.load(pickle_in)
constTable = pickle.load(pickle_in)

def main(argv):

    # ===== INICIAR MEMORIA =====
    LlenarMemoria(Memoria)

    # ===== POBLAR MEMORIA CONSTANTE =====
    for constantKey in constTable.constants.keys():
        memoryDir = constTable.constants[constantKey].memoryDir
        dataType = constTable.constants[constantKey].dataType
        if dataType == 'int':
            getMemorySection(memoryDir)[getStartingPoint(
                memoryDir)] = int(constantKey)
        elif dataType == 'char':
            getMemorySection(memoryDir)[
                getStartingPoint(memoryDir)] = constantKey
        elif dataType == 'bool':
            if constantKey == 'True':
                getMemorySection(memoryDir)[getStartingPoint(
                    memoryDir)] = True
            else:
                getMemorySection(memoryDir)[getStartingPoint(
                    memoryDir)] = False

    # ===== INICIAR EJECUCION =====
    i = 0

    while i < len(FilaQuadsMemoria):
        if FilaQuadsMemoria[i].operator == 'Goto':
            i = FilaQuadsMemoria[i].result - 1
            i = i - 1

        elif FilaQuadsMemoria[i].operator == 'GotoF':
            left = FilaQuadsMemoria[i].leftOp
            if not getMemorySection(left)[getStartingPoint(left)]:
                i = FilaQuadsMemoria[i].result - 1
                i = i - 1

        elif FilaQuadsMemoria[i].operator == '+':
            left = FilaQuadsMemoria[i].leftOp
            right = FilaQuadsMemoria[i].rightOp
            res = FilaQuadsMemoria[i].result

```

```

        getMemorySection(res)[getStartingPoint(res)] =
int(getMemorySection(left)[getStartingPoint(left)] +

getMemorySection(right)[getStartingPoint(right)])

        elif FilaQuadsMemoria[i].operator == '-':
            left = FilaQuadsMemoria[i].leftOp
            right = FilaQuadsMemoria[i].rightOp
            res = FilaQuadsMemoria[i].result
            getMemorySection(res)[getStartingPoint(res)] =
int(getMemorySection(left)[getStartingPoint(left)] -

getMemorySection(right)[getStartingPoint(right)])

        elif FilaQuadsMemoria[i].operator == '/':
            left = FilaQuadsMemoria[i].leftOp
            right = FilaQuadsMemoria[i].rightOp
            res = FilaQuadsMemoria[i].result
            getMemorySection(res)[getStartingPoint(res)] =
int(getMemorySection(left)[getStartingPoint(left)] /

getMemorySection(right)[getStartingPoint(right)])

        elif FilaQuadsMemoria[i].operator == '*':
            left = FilaQuadsMemoria[i].leftOp
            right = FilaQuadsMemoria[i].rightOp
            res = FilaQuadsMemoria[i].result
            getMemorySection(res)[getStartingPoint(res)] =
int(getMemorySection(left)[getStartingPoint(left)] *

getMemorySection(right)[getStartingPoint(right)])

        elif FilaQuadsMemoria[i].operator == '=':
            left = FilaQuadsMemoria[i].leftOp
            res = FilaQuadsMemoria[i].result
            getMemorySection(res)[getStartingPoint(res)] = getMemorySection(left)[
                getStartingPoint(left)]

        elif FilaQuadsMemoria[i].operator == '==':
            left = FilaQuadsMemoria[i].leftOp
            right = FilaQuadsMemoria[i].rightOp
            res = FilaQuadsMemoria[i].result
            if getMemorySection(left)[getStartingPoint(left)] ==
getMemorySection(right)[getStartingPoint(right)]:
                getMemorySection(res)[getStartingPoint(res)] = True
            else:
                getMemorySection(res)[getStartingPoint(res)] = False

        elif FilaQuadsMemoria[i].operator == '!=':
            left = FilaQuadsMemoria[i].leftOp
            right = FilaQuadsMemoria[i].rightOp
            res = FilaQuadsMemoria[i].result
            if getMemorySection(left)[getStartingPoint(left)] !=
getMemorySection(right)[getStartingPoint(right)]:

```

```

        getMemorySection(res)[getStartingPoint(res)] = True
    else:
        getMemorySection(res)[getStartingPoint(res)] = False

    elif FilaQuadsMemoria[i].operator == '<':
        left = FilaQuadsMemoria[i].leftOp
        right = FilaQuadsMemoria[i].rightOp
        res = FilaQuadsMemoria[i].result
        if getMemorySection(left)[getStartingPoint(left)] <
getMemorySection(right)[getStartingPoint(right)]:
            getMemorySection(res)[getStartingPoint(res)] = True
        else:
            getMemorySection(res)[getStartingPoint(res)] = False

    elif FilaQuadsMemoria[i].operator == '>':
        left = FilaQuadsMemoria[i].leftOp
        right = FilaQuadsMemoria[i].rightOp
        res = FilaQuadsMemoria[i].result
        if getMemorySection(left)[getStartingPoint(left)] >
getMemorySection(right)[getStartingPoint(right)]:
            getMemorySection(res)[getStartingPoint(res)] = True
        else:
            getMemorySection(res)[getStartingPoint(res)] = False

    elif FilaQuadsMemoria[i].operator == '&&':
        left = FilaQuadsMemoria[i].leftOp
        right = FilaQuadsMemoria[i].rightOp
        res = FilaQuadsMemoria[i].result
        if getMemorySection(left)[getStartingPoint(left)] and
getMemorySection(right)[getStartingPoint(right)]:
            getMemorySection(res)[getStartingPoint(res)] = True
        else:
            getMemorySection(res)[getStartingPoint(res)] = False

    elif FilaQuadsMemoria[i].operator == '||':
        left = FilaQuadsMemoria[i].leftOp
        right = FilaQuadsMemoria[i].rightOp
        res = FilaQuadsMemoria[i].result
        if getMemorySection(left)[getStartingPoint(left)] or
getMemorySection(right)[getStartingPoint(right)]:
            getMemorySection(res)[getStartingPoint(res)] = True
        else:
            getMemorySection(res)[getStartingPoint(res)] = False

    elif FilaQuadsMemoria[i].operator == '>':
        left = FilaQuadsMemoria[i].leftOp
        right = FilaQuadsMemoria[i].rightOp
        res = FilaQuadsMemoria[i].result
        if getMemorySection(left)[getStartingPoint(left)] >
getMemorySection(right)[getStartingPoint(right)]:
            getMemorySection(res)[getStartingPoint(res)] = True
        else:
            getMemorySection(res)[getStartingPoint(res)] = False

```



```

elif FilaQuadsMemoria[i].operator == 'print':
    left = FilaQuadsMemoria[i].leftOp
    print(getMemorySection(left)[getStartingPoint(left)])

elif FilaQuadsMemoria[i].operator == 'ERA':
    res = FilaQuadsMemoria[i].result
    createParamDict(res)
    # print(Memoria["Local"])
    # mandarDormir()
    # print(PilaDormir.top())
    pass

elif FilaQuadsMemoria[i].operator == 'GOSUB':
    left = FilaQuadsMemoria[i].leftOp
    PilaIndex.push(i + 1)
    i = functionDirectory.getStartPosition(left) - 2
    mandarDormir()
    reiniciarMemoria()
    insertParameters()
    pass

elif FilaQuadsMemoria[i].operator == 'EndFunc':
    i = PilaIndex.pop() - 1
    PilaParam.pop()
    mandarDespertar()

elif FilaQuadsMemoria[i].operator == 'PARAMETER':
    left = FilaQuadsMemoria[i].leftOp
    ParamList.append((left, getMemorySection(
        left)[getStartingPoint(left)]))

elif FilaQuadsMemoria[i].operator == 'RETURN':
    res = FilaQuadsMemoria[i].result
    PilaDir.push(res)
    i = PilaIndex.pop() - 1
    PilaParam.pop()
    # print(Memoria["Local"])

elif FilaQuadsMemoria[i].operator == '=*':
    res = FilaQuadsMemoria[i].result
    functionResult = PilaDir.pop()
    getMemorySection(res)[getStartingPoint(res)] = getMemorySection(
        functionResult)[getStartingPoint(functionResult)]
    mandarDespertar()

elif FilaQuadsMemoria[i].operator == 'Ver':
    left = FilaQuadsMemoria[i].leftOp
    right = FilaQuadsMemoria[i].rightOp
    res = FilaQuadsMemoria[i].result
    if getMemorySection(left)[getStartingPoint(left)] < right or
getMemorySection(left)[getStartingPoint(left)] >= res:
        raise Exception("No esta en los limites el Arreglo")

elif FilaQuadsMemoria[i].operator == 'pintar':

```

```

        MatColor.Paint()
        pass

    elif FilaQuadsMemoria[i].operator == 'mover':
        left = FilaQuadsMemoria[i].leftOp
        MatColor.Mover(getMemorySection(left)[getStartingPoint(left)])
        pass

    elif FilaQuadsMemoria[i].operator == 'cambiar':
        left = FilaQuadsMemoria[i].leftOp
        MatColor.Cambiar(left)
        pass

    elif FilaQuadsMemoria[i].operator == 'Exit':
        MatColor.printMat()
        MatColor.translate()
        print(MatColor.context)
        with open("MatColors.pickle", "wb") as f:
            pickle.dump(MatColor.context, f)

i = i + 1

```

c. SemanticCube.py

El cubo semántico lo utiliza el archivo listener.py para confirmar que las operaciones aritméticas son válidas. Se maneja la siguiente estructura:

```

class Operations:
    SUMA = "+"
    RESTA = "-"
    MULT = "*"
    DIVISION = "/"
    Y_SIMBOLO = "&&"
    O_SIMBOLO = "||"
    IGUAL = "=="
    MAYOR = ">"
    MENOR = "<"
    DIFERENTE = "!="
    ASIGNAR = "="

class Types():
    CHAR = "char"
    INT = "int"
    ERROR = "error: types do not match"
    BOOL = "bool"

class SemanticCube:
    cube = {
        # ===== INT == INT =====
        (Types.INT, Types.INT, Operations.SUMA): Types.INT,
        (Types.INT, Types.INT, Operations.RESTA): Types.INT,
        (Types.INT, Types.INT, Operations.DIVISION): Types.INT,
        (Types.INT, Types.INT, Operations.MULT): Types.INT,
    }

```

```

        (Types.INT, Types.INT, Operations.Y_SIMBOLO): Types.ERROR,
        (Types.INT, Types.INT, Operations.O_SIMBOLO): Types.ERROR,
        (Types.INT, Types.INT, Operations.IGUAL): Types.BOOL,
        (Types.INT, Types.INT, Operations.MAYOR): Types.BOOL,
        (Types.INT, Types.INT, Operations.MENOR): Types.BOOL,
        (Types.INT, Types.INT, Operations.DIFERENTE): Types.BOOL,
        (Types.INT, Types.INT, Operations.ASIGNAR): Types.INT,
    ...

```

d. Function.py

En el archivo listener.py se utilizan las funciones identificadas en este archivo para manejar las acciones que se pueden realizar en relación a una función.

```

def setStartPosition(self, funcName, startPosition):
    self.dictionary[funcName].startPosition = startPosition

def getStartPosition(self, funcName):
    return self.dictionary[funcName].startPosition
def exist(self, funcName):
    if funcName not in self.dictionary:
        raise Exception(
            "{} does not exist in directory".format(funcName))
    return True

def getNextInt(self, funcName):
    return self.dictionary[funcName].memory.getEntera()

def getNextBool(self, funcName):
    return self.dictionary[funcName].memory.getBooleanos()

def getNextChar(self, funcName):
    return self.dictionary[funcName].memory.getChar()

def getNextTemp(self, funcName):
    return self.dictionary[funcName].memory.getTemporales()

```

Para crear una función se maneja la siguiente estructura:

```

def add_function(self, funcName):
    if funcName in self.dictionary:
        raise Exception("The function already exists.")
    else:
        self.dictionary[funcName] = Function(
            funcName)

```

Una función maneja las siguientes características:

```

from antlr4 import *
from Variable import Variable
from memory import memory

```

```
# Store information about a function

class Function:
    def __init__(self, funcName):
        self.funcName = funcName
        self.startPosition = 0
        self.returnType = None
        self.memory = memory(4000, 5000, 6000, 0)
        self.parameters = []
        self.localVariables = 0
        self.temporalVariables = 0
        self.intParameters = 0
        self.boolParameters = 0
        self.charParameters = 0
```

e. Variable.py

En el archivo listener.py se utilizan las funciones identificadas en este archivo para manejar las acciones que se pueden realizar en relación a una variable.

```
def exist(self, varName):
    if varName not in self.vars.keys():
        raise Exception(
            "{} does not exist in the directory".format(varName))

def setLSup(self, varName, limite):
    self.vars[varName].node.superior = limite

def getLSup(self, varName):
    return self.vars[varName].node.superior

def get_local_variable(self, varName):
    return varName in self.vars.keys()

def setIsArray(self, varName):
    self.vars[varName].isArray = True

def add_variable(self, varName, dataType, scope, memoryDir):
    if varName in self.vars:
        raise Exception(
            "{} already exists in the directory".format(varName))

    elif varName in self.keywords:
        raise Exception("{} is a reserved word".format(varName))

    else:
        var = Variable(varName, dataType, scope, memoryDir)
        self.vars[varName] = var
```

Una variable maneja las siguientes características:

```
from antlr4 import *
```

```
# Store the variable table structure

class arrayNode:
    def __init__(self):
        self.inferior = 0
        self.superior = 0
        self.k = 0

class Variable:
    def __init__(self, varName, dataType, scope, memoryDir):
        self.varName = varName
        self.dataType = dataType
        self.scope = scope
        self.memoryDir = memoryDir
        self.isArray = False
        self.node = arrayNode()
```

f. memory.py

En este archivo se manejan los contadores de cada tipo de variable y se identifican las funciones para obtener las siguientes direcciones disponibles.

```
class memory:

    def __init__(self, enteras, booleanos, chars, temporales):
        self.enteras = enteras
        self.chars = chars
        self.booleanos = booleanos
        self.temporales = temporales

        self.i = 0
        self.b = 0
        self.c = 0
        self.t = 0

    def getEntera(self):
        self.i = self.i + 1
        return self.enteras + self.i - 1

    def getBooleanos(self):
        self.b = self.b + 1
        return self.booleanos + self.b - 1

    def getChar(self):
        self.c = self.c + 1
        return self.chars + self.c - 1

    def getTemporales(self):
        self.t = self.t + 1
        return self.temporales + self.t - 1

    def resetTemporales(self):
```

```
self.t = 0
```