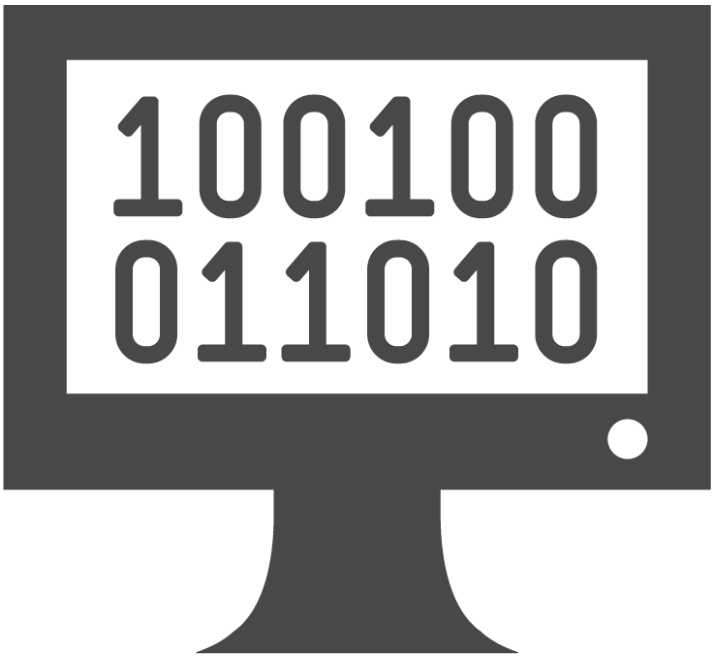
Diseño de Compiladores



Proyecto Final

2 de mayo de 2018

David de la Fuente Garza
A00817582

Barbara Valdez Mireles
A01175920

ONE FOR ALL

a). DESCRIPCIÓN DEL PROYECTO:

a.1) Propósito, Objetivos y Alcance del Proyecto.

El propósito del proyecto es el hacer un compilador orientado a objetos. El lenguaje soporta herencia simple y le permite al usuario crear clases con atributos privados y públicos; así mismo el lenguaje le permite al usuario crear arreglos de una dimensión.
Dentro de los objetivos se encuentra el manejo de ciclos, condiciones, funciones, arreglos y expresiones aritméticas.
El alcance del proyecto será desarrollar un compilador que genere los cuádruplos de ejecución y una máquina virtual que lea y ejecute estos cuádruplos.

a.2) Análisis de Requerimientos y Casos de Uso generales.

1. Los tipos de datos primitivos del proyecto deberán de ser BOOLEAN, FLOAT, INT y STRING.
2. El lenguaje debe de permitir al usuario hacer ciclos (WHILE) y condiciones simples (IF-ELSE).
3. El lenguaje debe de manejar entradas y salidas.
4. El lenguaje debe permitirle al usuario crear funciones.
5. El usuario deberá de escribir su código principal en una sección llamada main.
6. El usuario puede asignar valores a variables, siempre y cuando sean del mismo tipo de dato.
7. El usuario puede ejecutar expresiones aritméticas y lógicas.
8. El usuario puede igual el valor de una función a una variable.
9. El lenguaje permite el uso de variables locales y globales.
10. El lenguaje le permite al usuario crear clases y utilizarlas como un nuevo tipo de dato.
11. El lenguaje permite que las clases tengan atributos y métodos privados y públicos.
12. El usuario puede declarar objetos de la clase en el programa dentro de un contexto global o local.
13. El lenguaje permite hacer herencia simple.
14. El lenguaje le permite al usuario hacer colecciones de datos (ARREGLOS).

a.3) Descripción de los principales Test Cases.

# Prueba	Descripción de la prueba	Resultado
1	Fibonacci recursivo	Aprobada
Prueba utilizada para verificar que las funciones trabajan correctamente.		
2	Fibonacci iterativo	Aprobada
Prueba utilizada para verificar que los ciclos y condiciones funcionen		
3	Factorial recursivo	Aprobada
Prueba utilizada para verificar que las funciones trabajan correctamente.		
4	Factorial iterativo	

Prueba utilizada para verificar que los ciclos y condiciones funcionen		
5	Búsqueda en un arreglo	Aprobada
Prueba para verificar un manejo de arreglos de variables dimensionadas		
6	Bubble sort	Aprobada
Prueba para verificar un manejo de arreglos de variables dimensionadas		

a.4) Descripción del proceso

Date: Mon Apr 30 20:59:57 2018 -0500
Resolve merge conflicts

Date: Sun Apr 29 20:05:26 2018 -0500
Arithmetic statutes of virtual machine

Date: Sat Apr 21 01:33:30 2018 -0500
Merge pull request #6 from CharlieBradbury/Barbs_Compi
Add addresses to quadruples. Add the assignment quadruples

Date: Fri Apr 20 02:57:43 2018 -0500
Add addresses to quadruples. Add the assignment quadruples

Date: Sun Apr 15 02:05:24 2018 -0500
Generate IF-ELSE Quadruples

Date: Mon Apr 30 20:59:57 2018 -0500
Resolve merge conflicts

Date: Sun Apr 29 20:05:26 2018 -0500
Arithmetic statutes of virtual machine

Date: Sat Apr 21 01:33:30 2018 -0500
Merge pull request #6 from CharlieBradbury/Barbs_Compi
Add addresses to quadruples.
Add the assignment quadruples

Date: Fri Apr 20 02:57:43 2018 -0500
Add addresses to quadruples. Add the assignment quadruples

Date: Sun Apr 15 02:05:24 2018 -0500
Generate IF-ELSE Quadruples

Date: Mon Apr 30 20:59:57 2018 -0500
Resolve merge conflicts

Date: Sun Apr 29 20:05:26 2018 -0500
Arithmetic statutes of virtual machine

Date: Sat Apr 21 01:33:30 2018 -0500
Merge pull request #6 from CharlieBradbury/Barbs_Compi
Add addresses to quadruples. Add the assignment quadruples

Date: Fri Apr 20 02:57:43 2018 -0500
Add addresses to quadruples. Add the assignment quadruples

Date: Sun Apr 15 02:05:24 2018 -0500
Generate IF-ELSE Quadruples

Date: Fri Apr 13 10:04:24 2018 -0500
Update gitignore

Date: Wed Apr 11 16:11:46 2018 -0500
Arithmetic quadruple

Date: Mon Apr 30 20:59:57 2018 -0500
Resolve merge conflicts

Date: Sun Apr 29 20:05:26 2018 -0500
Arithmetic statutes of virtual machine

Date: Sat Apr 21 01:33:30 2018 -0500
Merge pull request #6 from CharlieBradbury/Barbs_Compi
Add addresses to quadruples. Add the assignment quadruples

Date: Fri Apr 20 02:57:43 2018 -0500
Add addresses to quadruples. Add the assignment quadruples

Date: Sun Apr 15 02:05:24 2018 -0500
Generate IF-ELSE Quadruples

Date: Fri Apr 13 10:04:24 2018 -0500
Update gitignore

Date: Wed Apr 11 16:11:46 2018 -0500
Merge pull request #5 from CharlieBradbury/OFA_David
Arithmetic quadruples

Date: Mon Apr 30 20:59:57 2018 -0500
Resolve merge conflicts

Date: Sun Apr 29 20:05:26 2018 -0500
Arithmetic statutes of virtual machine

Date: Sat Apr 21 01:33:30 2018 -0500
Merge pull request #6 from CharlieBradbury/Barbs_Compi
Add addresses to quadruples. Add the assignment quadruples

Date: Fri Apr 20 02:57:43 2018 -0500
Add addresses to quadruples. Add the assignment quadruples

Date: Sun Apr 15 02:05:24 2018 -0500
Generate IF-ELSE Quadruples

Date: Fri Apr 13 10:04:24 2018 -0500
Update gitignore

Date: Wed Apr 11 16:11:46 2018 -0500
Merge pull request #5 from CharlieBradbury/OFA_David
Arithmetic quadruples

Date: Wed Apr 11 16:10:52 2018 -0500
Merge branch 'master' into OFA_David

Date: Wed Apr 11 16:04:44 2018 -0500
Arithmetic quadruples

Date: Mon Apr 9 21:37:30 2018 -0500
QuadruplesD

Date: Fri Apr 6 23:07:08 2018 -0500
Change project structure

Date: Fri Apr 6 17:13:04 2018 -0500
Merge pull request #4 from CharlieBradbury/OFA_David
Merging Variables and Functions Directories into master

Date: Fri Apr 6 17:12:41 2018 -0500
Merge branch 'master' into OFA_David

Date: Fri Apr 6 16:47:25 2018 -0500
Variables and functions directory working

Date: Wed Apr 4 19:22:43 2018 -0500
Correct function directory issues

Date: Wed Apr 4 19:03:12 2018 -0500
Refactor code

Date: Wed Apr 4 18:59:35 2018 -0500
Moving content from inside folder to outside

Date: Wed Apr 4 18:58:36 2018 -0500
Modifying printing of classes information

Date: Sun Apr 1 00:53:04 2018 -0600
Modify grammar and create directory

Date: Sun Mar 25 15:04:11 2018 -0600
Add directory manager. Add class, function, and variable objects

Date: Mon Mar 5 14:55:06 2018 -0600
Fixed issue with commas in parameters

Date: Sun Mar 4 20:14:10 2018 -0600
Avance parser y scanner

Date: Sun Mar 4 01:56:10 2018 -0600
Avance parser y scanner

Reflexión

Hacer un compilador implica escribir muchas líneas de código. Al realizar este proyecto, aprendí la importancia de modular funcionalidades para poderlas reusar. Además dado que es un proyecto de mucha complejidad es importante tener un buen diseño, desde un principio lo cual creo que pudimos haber hecho un mejor trabajo, pero me queda como experiencia el tener un buen diseño antes de empezar a codificar, ya que tuve que refactorizar mi código en múltiples ocasiones para hacerlo más limpio.

Es uno de los proyectos más largos en los que he trabajado, y tuve una satisfacción muy grande al ver que el compilador funcionaba.

_____

Barbara Valdez

Reflexión

Realizar un compilador me ayudó para valorar mejor el esfuerzo que involucra hacer un lenguaje de programación y su compilación. Definitivamente no es una tarea trivial, pero sí una muy interesante y que pone en práctica amplios conocimiento de la carrera en general. Me gustó mucho que fuera un proceso que se fuera refinando, buscando siempre hacer mejoras desde el diseño de la gramática hasta la elaboración de la máquina virtual.

Ver finalmente compilar el código y poder hacer propios programas es una experiencia muy placentera porque recompensa todo el trabajo de abstracción realizado. El apoyo de mi pareja siempre fue de gran ayuda y este proyecto no pudo haber salido adelante sin el trabajo de ambos.

_____

David de la Fuente

b). DESCRIPCION DEL LENGUAJE:

b.1) Nombre del lenguaje.

One For All

b.2) Descripción genérica de las principales características del lenguaje

Es un lenguaje orientado a objetos que permite herencia simple. Está desarrollado en Python y se utiliza ANTLR como herramienta para generar el analizador sintáctico y gramática.

Nuestro lenguaje le permite al usuario declarar clases, variables y funciones. Además, le permite ejecutar expresiones aritméticas, lógicas y relacionales.

Las palabras reservadas de nuestro lenguaje son:

program	class	public	private	var
int	string	bool	write	read
function	return	if	else	while
main	init	True	False	

Los caracteres especiales de nuestro lenguaje son:

=	(	)	>	<
>=	<=	+	-	*
/	[	]	“ “	{ }

Para inicializar un programa

```
program test01;
```

Declarar una clase

```
class nombreClase : clasePadre {
	public var int variablePublica;
	private var string variablePrivada;

	private function string metodoPrivado(var string variablePrivada, var int
variablePublica) {

		}
}
```

Declarar variables globales

```
var int globalnt;
var float globalFloat;
var string globaString;
var bool globalBool;

//También se pueden declarar variables de tipo objeto, siempre y cuando ya se haya
declarado la clase.
var nombreClase globalClase;
```

Declaración de funciones

```
function string funcion(var string param) {
	var int varLocal = 0;
	globalnt = 4 * 3 + 8 * 9;
	return param;
}
```

Declaración de la función principal

```
main {
	//Declaración de variables locales y asignaciones
	var string python = "hello";
```

```
        var int hello = 2, hi = 4, bye = 5;

        //Declaración de arreglos
        var int iArr[2];

        //Asignar valor a una variable global bool
        j = (1 * (2 + 4) * 6 + 10) > 50;

        // Condiciones
        if(3 > 2) {
                barbara = 3 + 2;
               //Salida
                write("Hello World");
        } else {
                barbara = 3 * 4;
        }
        //Retorno que indica la finalización del contexto
        return 0;
}
```

b.3) Listado de los errores que pueden ocurrir, tanto en compilación como en ejecución.

Tipo de Error	Descripción
Variable not defined	La variable se intentó acceder pero no se encontró en el directorio.
Error while creating variable	Hubo algún problema en el proceso de creación de variable, ya sea en el número de parámetros o en los tipos de los mismos
Variable already declared	Se intenta declarar una variable que ya fue previamente declarada. No hay problema en múltiples asignaciones, sólo múltiples declaraciones.
Invalid Operation	Se intenta hacer una operación con operandos de tipos no válidos según el cubo semántico (Por ejemplo, sumar un string y un entero).
Type mismatch	Utilizados para condicionales, ocurre cuando el tipo no es el recibido (por ejemplo, al esperar recibir un booleano como resultado de un condición).
Out of bounds	Acceso fuera del rango de la memoria definida para un arreglo.
Class does not exist	Intento de instanciar o acceder a contenidos de una clase que no ha sido declarada.
Function not defined	La variable se intentó acceder pero no se encontró en el directorio.
Error while creating function	Hubo algún problema en el proceso de creación de la función, ya sea en el número de parámetros o en los tipos de los mismos
Function already declared	Se intenta declarar una función que ya fue previamente declara.

c). DESCRIPCION DEL COMPILADOR:

c.1) Equipo de cómputo, lenguaje y utilerías especiales usadas en el desarrollo del proyecto.

El compilador se desarrolló en Python y se hizo uso del framework Antlr para la generación del analizador de sintaxis. Además de estas herramientas, se usaron librerías de sys y os que son propias de Python.

El compilador se desarrolló en MacOS y Windows 10, también se utilizaron los editores de texto Visual Studio Code y Sublime Text durante la fase de desarrollo.

c.2) Descripción del Análisis de Léxico:

Patrones de Construcción

Simple digit: [0-9]+
Real numbers: [0-9]+('.'[0-9]+)
Strings: '"'.*?'"'
Letters: [a-zA-Z]+

Tokens del lenguaje

```
STRING : '"'.*?'"';
FLOAT : [0-9]+('.'[0-9]+);
INT : [0-9]+;
BOOLEAN : TRUE | FALSE;
TOK_AND : '&&' ;
TOK_OR  : '||' ;
TRUE : 'true' ;
FALSE : 'false';
TOK_IF : 'if';
TOK_ELSE : 'else';
TOK_WHILE : 'while';
TOK_VAR : 'var';
TOK_PROGRAM : 'program';
TOK_CLASS : 'class';
TOK_PRIVATE : 'private';
TOK_PUBLIC : 'public';
TOK_MAIN : 'main';
TOK_READ : 'read';
TOK_FUNCTION : 'function';
TOK_WRITE :'write';
TOK_LPAREN : '(';
TOK_RPAREN : ')';
TOK_LBRACE : '{';
TOK_RBRACE : '}';
TOK_LBRACKET : '[';
TOK_RBRACKET : ']';
TOK_PLUS : '+';
TOK_MINUS : '-';
TOK_MULTIPLICATION : '*';
TOK_DIVISION : '/';
TOK_EQUAL : '=';
TOK_DIFFERENT : '!=';
TOK_GREATER : '>';
TOK_LESS : '<';
TOK_GREATER_EQ : '>=';
TOK_LESS_EQ : '<=';
TOK_SAME : '==';
TOK_SEMICOLON : ';';
TOK_COLON : ':';
TOK_DOT : '.';
TOK_COMMA : ',';
TOK_INT : 'int';
TOK_FLOAT : 'float';
TOK_BOOLEAN : 'bool';
TOK_STRING : 'string';
TOK_RETURN : 'return';
TOK_ID : [a-zA-Z0-9]+;
```

c.3) Descripción del Análisis de Sintaxis

```
programa:
        TOK_PROGRAM TOK_ID TOK_SEMICOLON (classes)? (variables)? neuro_jump_main
(routines)? restOfProgram;

neuro_jump_main:
;

restOfProgram:
   main;
```

```
classes:
   (class_definition)+;

class_definition:
        TOK_CLASS TOK_ID inheritance TOK_LBRACE (class_public)? (class_private)? constructor
TOK_RBRACE;

inheritance:
   (TOK_COLON TOK_ID)?;

class_public:
   (TOK_PUBLIC variables | TOK_PUBLIC routines)+;

class_private:
    (TOK_PRIVATE variables | TOK_PRIVATE routines)+;

constructor:
        TOK_INIT TOK_LPAREN (parameters)? TOK_RPAREN block;

routines:
   (routine_definition)+;

routine_definition:
   TOK_FUNCTION data_type TOK_ID TOK_LPAREN (parameters)? TOK_RPAREN block;

parameters:
   TOK_VAR data_type TOK_ID (TOK_LBRACKET expressions neuro_array TOK_RBRACKET)?
parameters_recursive;

parameters_recursive:
   (TOK_COMMA  TOK_VAR neuroparam_rec data_type TOK_ID (TOK_LBRACKET expressions
neuro_array TOK_RBRACKET)?)*;

neuro_array:
;

neuroparam_rec:
;

variables:
   (variable_definition | variable_assign)+;

variable_definition:
        TOK_VAR data_type TOK_ID (TOK_LBRACKET expressions TOK_RBRACKET)? (TOK_COMMA
TOK_ID (TOK_LBRACKET expressions TOK_RBRACKET)? )* TOK_SEMICOLON;

variable_assign:
        TOK_VAR data_type TOK_ID (TOK_LBRACKET expressions TOK_RBRACKET)? TOK_EQUAL
expressions (TOK_COMMA TOK_ID (TOK_LBRACKET expressions TOK_RBRACKET)? TOK_EQUAL
expressions)* TOK_SEMICOLON;

data_type:
   (TOK_INT | TOK_FLOAT | TOK_STRING | TOK_BOOLEAN | TOK_ID);
```

```
main:
   TOK_MAIN block;

block:
  TOK_LBRACE statute TOK_RBRACE;

return_expr:
            TOK_RETURN expressions TOK_SEMICOLON;

statute:
   (assignment | condition | loop | output | input_ | variables |return_expr)*;

assignment:
             id_ TOK_EQUAL expressions TOK_SEMICOLON;

condition:
   TOK_IF TOK_LPAREN expressions TOK_RPAREN neuro_if block condition_else neuro_endif;

neuro_if:
;

neuro_endif:
;

loop:
   TOK_WHILE neuro_while_begin TOK_LPAREN expressions TOK_RPAREN neuro_while_expression
block neuro_while_end;

neuro_while_begin:
;

neuro_while_expression:
;

neuro_while_end:
;

input_:
   TOK_READ TOK_LPAREN expressions TOK_COMMA TOK_ID TOK_RPAREN TOK_SEMICOLON;

output:
            TOK_WRITE TOK_LPAREN expressions neuro_getOutput (output_recursive)?
neuro_finishOutput TOK_RPAREN TOK_SEMICOLON;

output_recursive:
            (TOK_COMMA expressions neuro_getOutput)+;

neuro_getOutput:
;

neuro_finishOutput:
;
```

```
condition_else:
   (neuro_else TOK_ELSE block)?;

neuro_else:
;

expressions:
        (expression_definition)+;

expression_definition:
   relational_exprs ((token_and | token_or) relational_exprs neuro_expression)?;

neuro_expression:
        ;

token_and:
        TOK_AND;

token_or:
        TOK_OR;

relational_exprs:
        (relational_expr_definition)+;

relational_expr_definition:
   sumMinus_exprs ((token_same | token_different | token_greater | token_greater_eq | token_less |
token_less_eq) sumMinus_exprs neuro_relational)?;

neuro_relational:
        ;

token_same:
        TOK_SAME;

token_different:
        TOK_DIFFERENT;

token_greater:
        TOK_GREATER;

token_greater_eq:
        TOK_GREATER_EQ;

token_less:
        TOK_LESS;

token_less_eq:
        TOK_LESS_EQ;

sumMinus_exprs:
        (sumMinus_expr_definition)+;

sumMinus_expr_definition:
   multiDiv_exprs neuro_sumMinus (token_plus | token_minus)?;
```

```
neuro_sumMinus:
        ;

token_plus:
        TOK_PLUS;

token_minus:
        TOK_MINUS;

multiDiv_exprs:
        (multiDiv_expr_definition)+;

multiDiv_expr_definition:
   factor neuro_multiDiv (token_multiplication | token_division)?;

neuro_multiDiv:
        ;

token_multiplication:
        TOK_MULTIPLICATION;

token_division:
        TOK_DIVISION;

factor:
   (token_lparen expressions token_rparen) | constant;

token_lparen:
        TOK_LPAREN;

token_rparen:
        TOK_RPAREN;

constant:
        (FLOAT | INT | STRING | BOOLEAN | id_);

id_:
        (id_definition_)+;

id_definition_:
        TOK_ID | evaluate_class | evaluate_function | evaluate_array | init_class;

init_class:
        TOK_ID TOK_EQUAL TOK_INIT TOK_LPAREN (expressions neuro_initEval (TOK_COMMA)?)*
neuro_createConstructor TOK_RPAREN TOK_SEMICOLON;

neuro_initEval:
;

neuro_createConstructor:
;

evaluate_class:
```

TOK_ID TOK_DOT TOK_ID(TOK_LPAREN expressions TOK_RPAREN)?;

evaluate_function:
TOK_ID TOK_LPAREN (expressions (TOK_COMMA)?)* neuro_params TOK_RPAREN;

neuro_params:
;

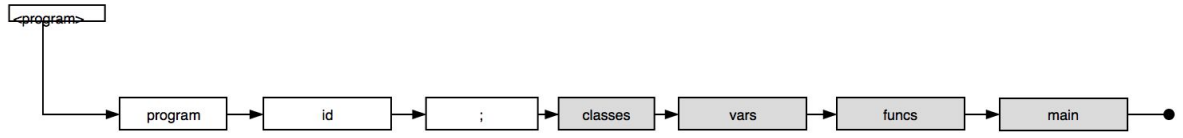
evaluate_array:
TOK_ID TOK_LBRACKET expressions TOK_RBRACKET;

c.4) Descripción de Generación de Código Intermedio y Análisis Semántico.

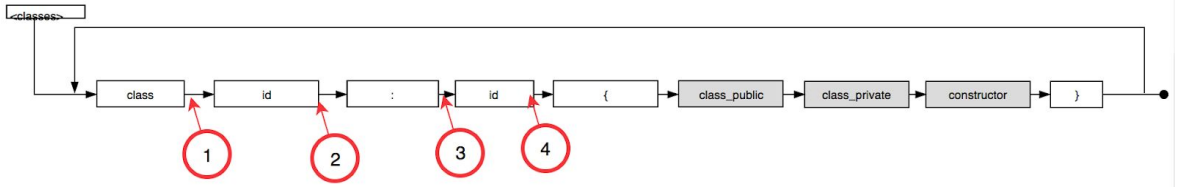
Operación	Código de Operación	Propósito
=	0	Operador aritmético
+	1	Operador aritmético
-	2	Operador aritmético
*	3	Operador aritmético
/	4	Operador aritmético
>	5	Operador relacional
>=	6	Operador relacional
<	7	Operador relacional
<=	8	Operador relacional
==	9	Operador relacional
!=	10	Operador relacional
&&	11	Operador lógico
||	12	Operador lógico
GOTOF	13	Condicionales y ciclos
GOTO	14	Saltos
GOSUB	15	Funciones
PARAM	16	Pase de parámetros
RETURN	17	Valor de retorno
ERA	18	Inicializa funciones
RETURN_ASSIGN	19	Asignar el valor de retorno
READ	20	Entrada
WRITE	21	Salida
END	22	Finalización del main
END_WRITE	23	Finalización de cuádruplos de write

ARRAY_DECLARE	24	Construcción de arreglos
ARRAY_POS	25	Manejo de posiciones de arreglo
INIT_CLASS	25	Inicialización de un objeto
INHERITS	26	Herencia simple
END_CLASS	27	Finalización de una clase

Diagramas de Sintaxis

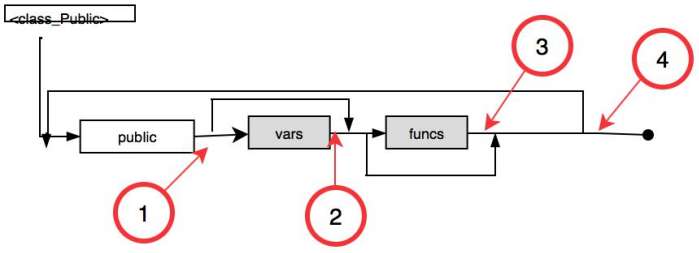


En este diagrama de sintaxis se especifica la estructura general del programa. Como se puede observar las clases, variables y funciones son opcionales, no obstante se debe de respetar el orden anterior.

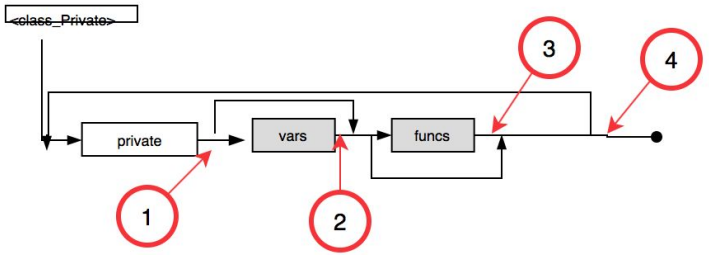


Cuando entra a la regla de classes:

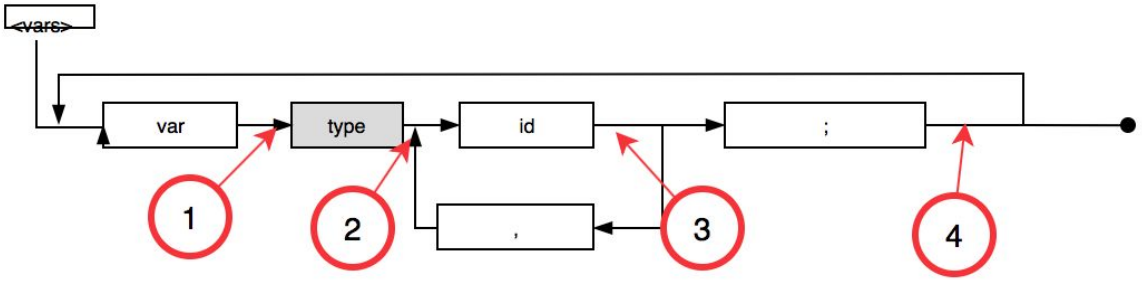
1) Lee que es una clase
2) Registra el nombre de la clase
3) Se prepara para recibir herencia, crea una variable para almacenar el nombre del padre
4) Asigna el valor de id en la variable del padre
   Crear una objeto objClass con el nombre de la clase y el padre
   Registrar variable en el directorio de clases



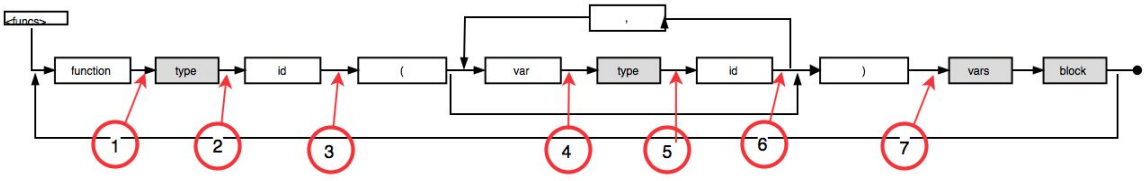
1. Se prepara una variable global que indique las siguientes variables o funciones son públicas
2. Después de salir de vars o funcs va a tener las variables o funciones que son públicas y hay que almacenarlas.



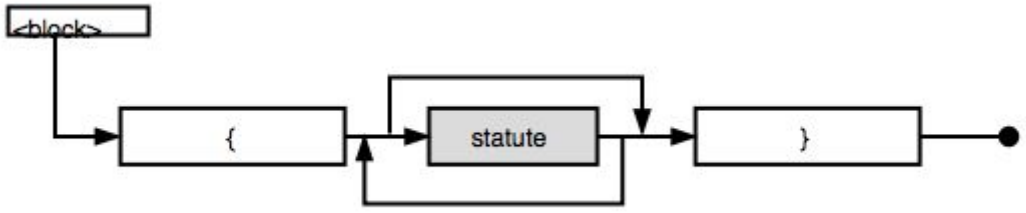
1. Se prepara una variable global que indique las siguientes variables o funciones son privadas
2. Después de salir de vars o funcs va a tener las variables o funciones que son privadas y hay que almacenarlas.



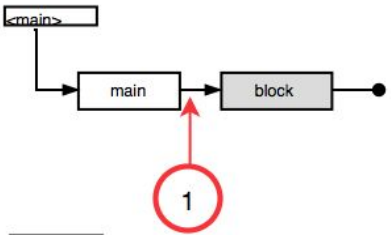
Al entrar a la declaración de variables, se debe de guardar el tipo y el id el cual puede ser un arreglo o un TOK_ID simple y al finalizar se crea un objVariable para almacenarse en el directorio de variables.



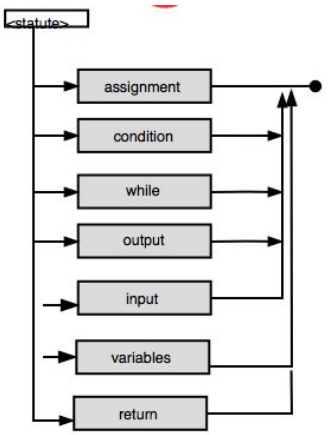
Al entrar a la declaración de función se identifica el tipo, el nombre, y se registran los parámetros de la misma. Al terminar de leer los parámetros se registra la función en el directorio.



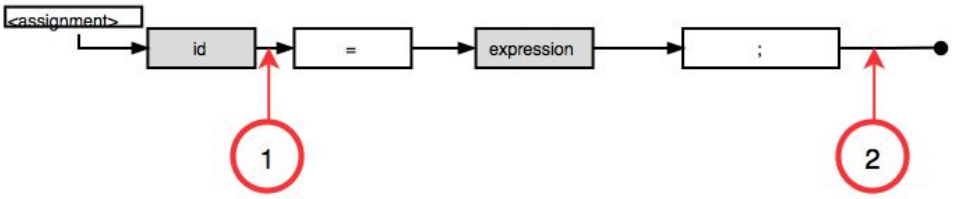
Un bloque es un conjunto de múltiples estatutos.



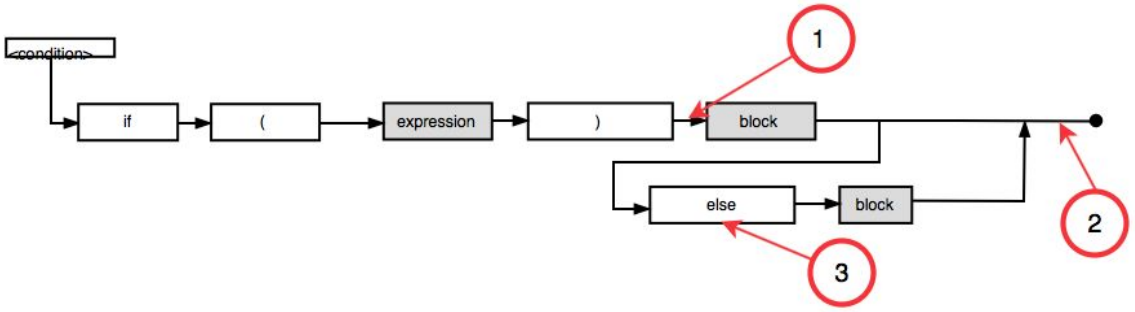
Después de leer el main se debe hacer el salto Goto para que sea la primera instrucción en ejecutarse después de las asignaciones globales.



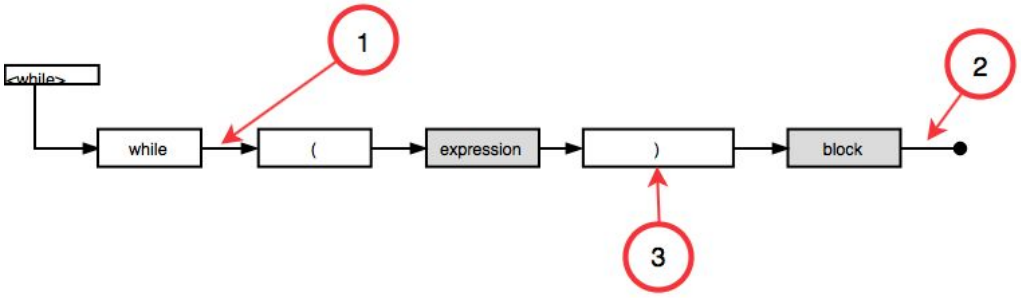
Un estatuto puede ser ya sea una asignación, una condición, una declaración, etc.



En la asignación se detecta el ID al que se quiere asignar, al llegar al punto 2 se asigna la expresión al id identificado en el paso 1.

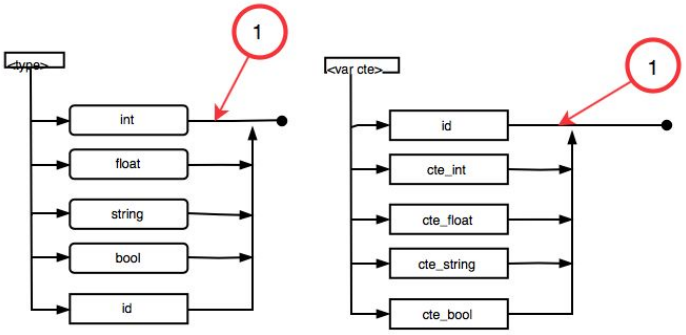


Cuando hay una condición se debe verificar que la condición sea boolean y se genera un cuádruplo GotoF el cual se llenará cuando se llegue al else o al final del if en caso de no haber else.

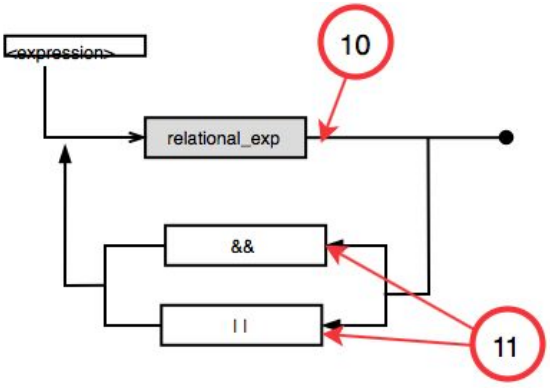


Al leer la palabra while, se guarda un registro de a donde se debe regresar. Si no se cumple la condición, se

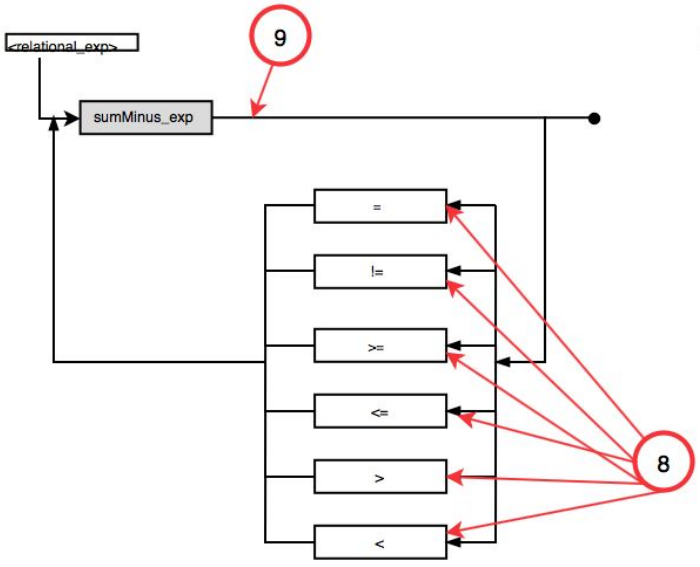
hace un salto al punto 2. En el punto 3 se hace un salto inmediato. la condición en el punto 1.



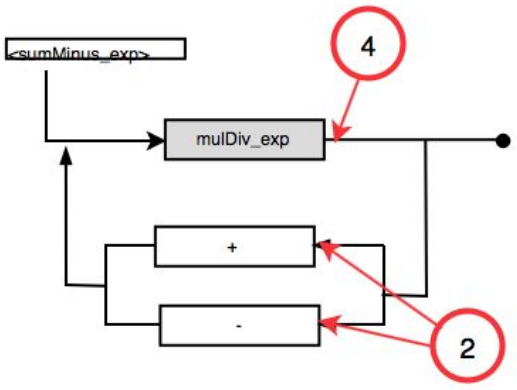
En el punto 1 del diagrama izquierdo se prepara una nueva variable del tipo que acaba de ser leído. Similar en el caso del diagrama derecho.



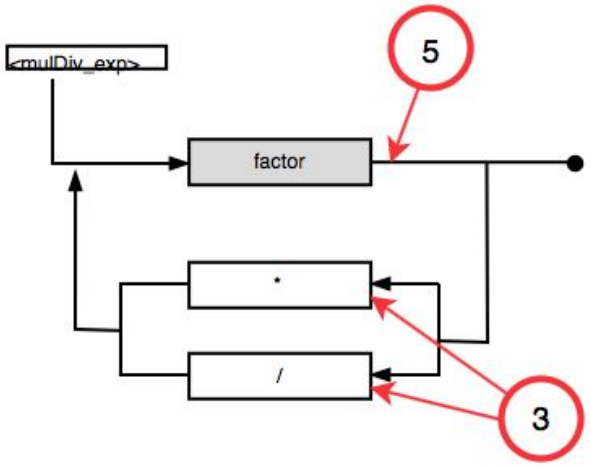
Al terminar de leer una expresión relacional, se revisa en la pila de operadores hay una operación de prioridad pendiente, si la hay se resuelve. En el punto 11 se introduce el operador que haya sido leído



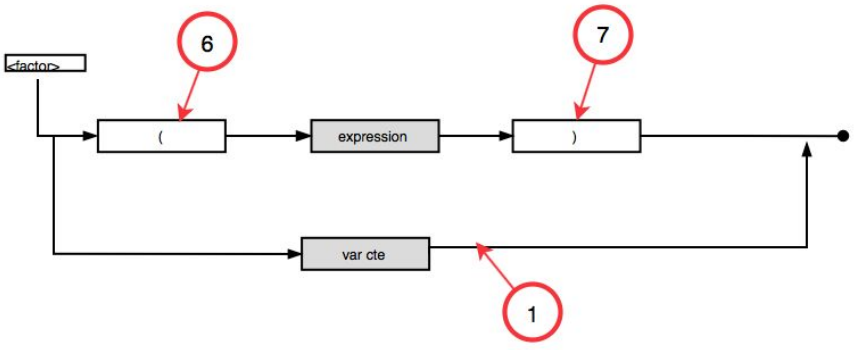
Al leer una expresión relacional, en el punto 9 se revisa si lo que está en el top de la pila es ya sea una suma o resta , si lo es, la resuelve. En el punto 8 se mete el símbolo a la pila de operandos.



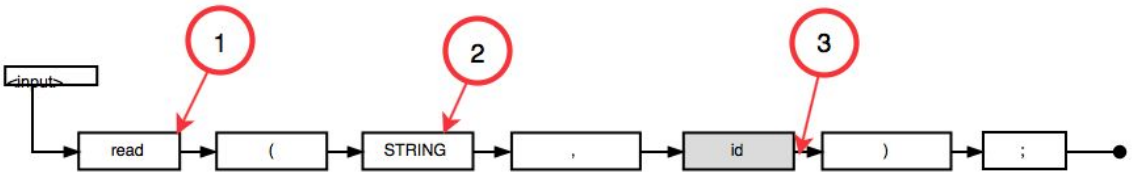
Al leer una expresión relacional, en el punto 4 se revisa si lo que está en el top de la pila es ya sea una suma o resta,, si lo es, la resuelve. En el punto 9 se mete el símbolo a la pila de operandos.



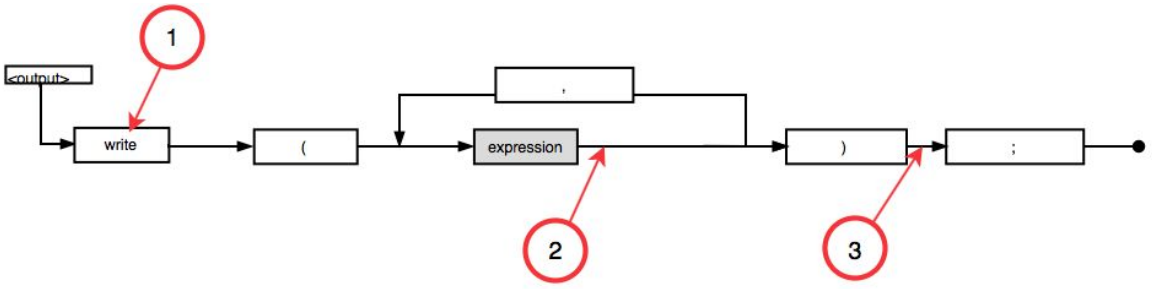
Al leer una expresión relacional, en el punto 5 se revisa si lo que está en el top de la pila es ya sea una multiplicación o división , si lo es, la resuelve. En el punto 3 se mete el símbolo a la pila de operandos.



Al llegar al punto 6 se introduce un piso falso de paréntesis, el cual separa las expresiones siguientes en otro nivel. En el punto 7 se quita el piso falso. En el piso 1 se registra la constante leída.



Se detecta que viene un read, entonces se espera un string el cual será asignado a una variable de tipo string.



Al hacer un output se crean dos tipos de cuádruplos el WRITE y el END_WRITE, debido a su naturaleza de imprimir varios elementos anidados. Cada que llegue al punto (2) se creará el cuádruplo de Write y al llegar al (3) se crea un cuádruplo de END_WRITE. Esto para saber cuales elementos van en un mismo output.

c.5) Descripción detallada del proceso de Administración de Memoria usado en la compilación.
Para la administración de memoria durante la generación de cuádruplos se establecieron rangos de direcciones para la parte global, local y temporal. Dentro de cada uno de estos apartados se separó por tipos de datos primitivos: string, bool, int y float; y además se apartaron direcciones para tipo obj, para la administración de objetos.

En la siguiente tabla se muestran los rangos en los que se dividió la memoria.

Contexto	Tipo de dato	Direcciones asociadas
Global	int	1000 - 2999
	float	3000 - 4999
	bool	5000 - 6999
	string	7000 - 8999
	obj	9000 - 10999
Local	int	11000 - 12999
	float	13000 - 14999
	bool	15000 - 16999
	string	17000 - 18999
	obj	19000 - 20999
	int	23000 - 24999

Temporal	float	25000 - 26999
	bool	27000 - 28999
	string	29000 - 30999
	obj	31000 - 32999

Para el manejo de direcciones se desarrolló la clase **addressManager.py** para definir los rangos estáticos de cada tipo de variable y su contexto correspondiente. La función **getVirtualAddress** se creó para obtener una dirección disponible dentro del contexto actual y dependiendo del tipo de dato.. Se creó la función **updateVirtualAddress** dentro de la clase para actualizar los contadores al momento de crear una variable o utilizar una variable temporal. La función **restartVirtualAddress** se definió para reiniciar las variables temporales y locales cada que se cambia de contexto.
Por último, se tiene la función **getMemorySegment** para utilizarla en la máquina virtual y saber el tipo de dato y el contexto. Se implementó para parsear valores de los cuádruplos y guardarlas en el directorio correspondiente.

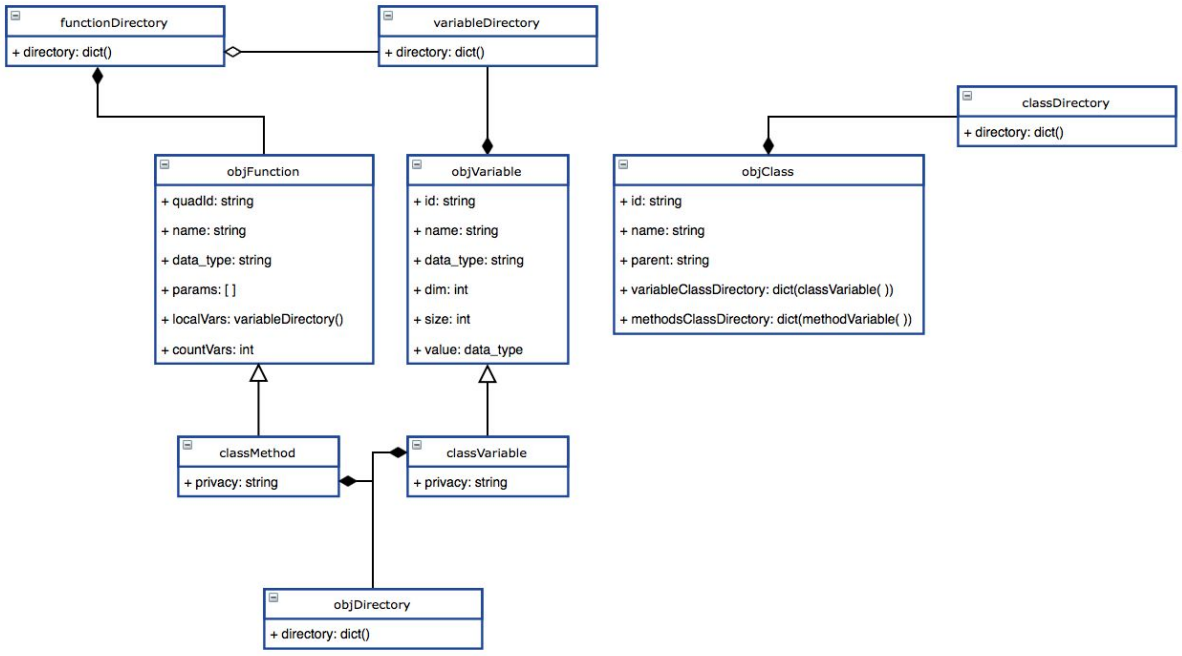
```
def getVirtualAddress(self, data_Type, scope):
        if scope == "temporal":
                if data_Type == "int":
                        return self.COUNTER_TEMPORALS_INT
                elif data_Type == "float":
                        return self.COUNTER_TEMPORALS_FLOAT
                elif data_Type == "bool":
                        return self.COUNTER_TEMPORALS_BOOLE
                elif data_Type == "string":
                        return self.COUNTER_TEMPORALS_STRIN
        elif scope == "local":
                if data_Type == "int":
                        return self.COUNTER_LOCALS_INT
                elif data_Type == "float":
                        return self.COUNTER_LOCALS_FLOAT
                elif data_Type == "bool":
                        return self.COUNTER_LOCALS_BOOLEAN
                elif data_Type == "string":
                        return self.COUNTER_LOCALS_STRING
        elif scope == "global":
                if data_Type == "int":
                        return self.COUNTER_GLOBALS_INT
                elif data_Type == "float":
                        return self.COUNTER_GLOBALS_FLOAT
                elif data_Type == "bool":
                        return self.COUNTER_GLOBALS_BOOLEAN
                elif data_Type == "string":
                        return self.COUNTER_GLOBALS_STRING
```

```
def updateVirtualAddress(self, data_Type, scope):
        if scope == "temporal":
                if data_Type == "int":
                        self.COUNTER_TEMPORALS_INT += 1
                elif data_Type == "float":
                        self.COUNTER_TEMPORALS_FLOAT += 1
                elif data_Type == "bool":
                        self.COUNTER_TEMPORALS_BOOLEAN += 1
                elif data_Type == "string":
                        self.COUNTER_TEMPORALS_STRING += 1
        elif scope == "local":
                if data_Type == "int":
                        self.COUNTER_LOCALS_INT += 1
                elif data_Type == "float":
                        self.COUNTER_LOCALS_FLOAT += 1
                elif data_Type == "bool":
                        self.COUNTER_LOCALS_BOOLEAN += 1
                elif data_Type == "string":
                        self.COUNTER_LOCALS_STRING += 1
        elif scope == "global":
                if data_Type == "int":
                        self.COUNTER_GLOBALS_INT += 1
                elif data_Type == "float":
                        self.COUNTER_GLOBALS_FLOAT += 1
                elif data_Type == "bool":
                        self.COUNTER_GLOBALS_BOOLEAN += 1
                elif data_Type == "string":
                        self.COUNTER_GLOBALS_STRING += 1
```

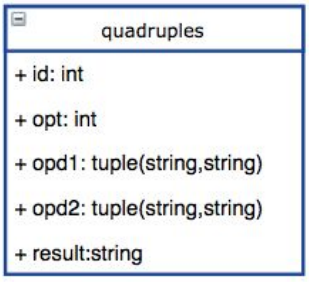
```
def restartVirtualAddress(self):
        self.COUNTER_TEMPORALS_INT = self.TEMPORALS_INT
        self.COUNTER_TEMPORALS_FLOAT = self.TEMPORALS_FLOAT
        self.COUNTER_TEMPORALS_BOOLEAN = self.TEMPORALS_BOOLEAN
        self.COUNTER_TEMPORALS_STRING = self.TEMPORALS_STRING
        self.COUNTER_TEMPORALS_OBJECT = self.TEMPORALS_OBJECT
        self.COUNTER_LOCALS_INT = self.LOCALS_INT
        self.COUNTER_LOCALS_FLOAT = self.LOCALS_FLOAT
        self.COUNTER_LOCALS_BOOLEAN = self.LOCALS_BOOLEAN
        self.COUNTER_LOCALS_STRING = self.LOCALS_STRING
        self.COUNTER_LOCALS_OBJECT = self.LOCALS_OBJECT
```

```
# Method that given an address direction, return to which segment belongs
def getMemorySegment(self, stringAddress):
        # Parse address to int
        address = int(stringAddress)

        # Context and varType variables
        context = "NoContext"
        varType = "NoType"

        if self.BEGIN_GLOBALS <= address <= self.END_GLOBALS:
                context = "global"
                if self.GLOBALS_INT <= address < self.GLOBALS_FLOAT:
                        varType = "int"
                elif self.GLOBALS_FLOAT <= address < self.GLOBALS_BOOLEAN:
                        varType = "float"
                elif self.GLOBALS_BOOLEAN <= address < self.GLOBALS_STRING:
                        varType = "boolean"
                elif self.GLOBALS_STRING <= address < self.GLOBALS_OBJECT:
                        varType = "string"
                elif self.GLOBALS_OBJECT <= address <= self.END_GLOBALS:
                        varType = "obj"
        elif self.BEGIN_LOCALS <= address <= self.END_LOCALS:
                context = "local"
                if self.LOCALS_INT <= address < self.LOCALS_FLOAT:
                        varType = "int"
                elif self.LOCALS_FLOAT <= address < self.LOCALS_BOOLEAN:
                        varType = "float"
                elif self.LOCALS_BOOLEAN <= address < self.LOCALS_STRING:
                        varType = "boolean"
                elif self.LOCALS_STRING <= address < self.LOCALS_OBJECT:
                        varType = "string"
                elif self.LOCALS_OBJECT <= address <= self.END_LOCALS:
                        varType = "obj"
        elif self.BEGIN_TEMPORALS <= address <= self.END_TEMPORALS:
                context = "temporal"
                if self.TEMPORALS_INT <= address < self.TEMPORALS_FLOAT:
                        varType = "int"
                elif self.TEMPORALS_FLOAT <= address < self.TEMPORALS_BOOLEAN:
                        varType = "float"
                elif self.TEMPORALS_BOOLEAN <= address < self.TEMPORALS_STRING:
                        varType = "boolean"
                elif self.TEMPORALS_STRING <= address < self.TEMPORALS_OBJECT:
                        varType = "string"
                elif self.TEMPORALS_OBJECT <= address <= self.END_TEMPORALS:
                        varType = "obj"
        else:
                self.error.definition(self.error.INVALID_MEMORY_ACCESS, stringAddress

        memorySegment = [context, varType]
        return memorySegment
```

Arquitectura de los directorios



Cuádruplos



d). DESCRIPCION DE LA MÁQUINA VIRTUAL:

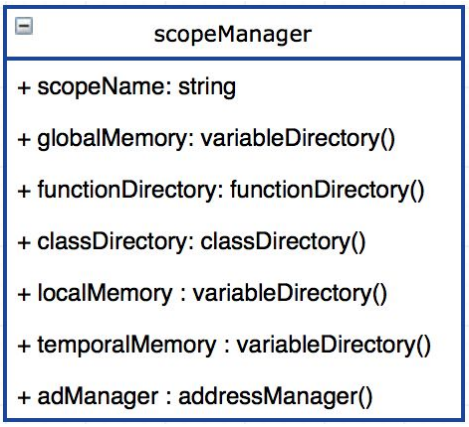
d.1) Descripción detallada del proceso de Administración de Memoria en ejecución

Para manejar los diversos scopes se utilizó la clase scopeManager, la cual tiene} principalmente tres tipos de memoria (global, local y temporal. los tres siendo directorios de variables). Se tomó a lo global como un contexto propio, mientras que el main y las funciones se tomaron como contextos locales. Todos los contextos tienen información almacenada en la memoria temporal (pues es la que almacena el resultado de las operaciones intermedias), sin embargo únicamente el contexto global utilizaba la memoria global mientras que los demás contextos la memoria local.

El proceso de saber en qué memoria debían de buscarse y almacenarse la información se hizo utilizando un método auxiliar getMemorySegment, el cual dada una dirección de memoria regresaba su contexto y su tipo de dato. Esto se hacía basándose en los rangos establecidos para el manejo de memoria.

```
# Method that given an address direction, return to which segment belongs
def getMemorySegment(self, stringAddress):
    # Parse address to int
    address = int(stringAddress)

    # Context and varType variables
    context = "NoContext"
    varType = "NoType"

    if self.BEGIN_GLOBALS <= address <= self.END_GLOBALS:
        context = "global"
        if self.GLOBALS_INT <= address < self.GLOBALS_FLOAT:
            varType = "int"
        elif self.GLOBALS_FLOAT <= address < self.GLOBALS_BOOLEAN:
            varType = "float"
        elif self.GLOBALS_BOOLEAN <= address < self.GLOBALS_STRING:
            varType = "boolean"
        elif self.GLOBALS_STRING <= address < self.GLOBALS_OBJECT:
            varType = "string"
        elif self.GLOBALS_OBJECT <= address <= self.END_GLOBALS:
            varType = "obj"
    elif self.BEGIN_LOCALS <= address <= self.END_LOCALS:
        context = "local"
        if self.LOCALS_INT <= address < self.LOCALS_FLOAT:
            varType = "int"
        elif self.LOCALS_FLOAT <= address < self.LOCALS_BOOLEAN:
            varType = "float"
        elif self.LOCALS_BOOLEAN <= address < self.LOCALS_STRING:
            varType = "boolean"
        elif self.LOCALS_STRING <= address < self.LOCALS_OBJECT:
            varType = "string"
        elif self.LOCALS_OBJECT <= address <= self.END_LOCALS:
            varType = "obj"
    elif self.BEGIN_TEMPORALS <= address <= self.END_TEMPORALS:
        context = "temporal"
        if self.TEMPORALS_INT <= address < self.TEMPORALS_FLOAT:
            varType = "int"
        elif self.TEMPORALS_FLOAT <= address < self.TEMPORALS_BOOLEAN:
            varType = "float"
        elif self.TEMPORALS_BOOLEAN <= address < self.TEMPORALS_STRING:
            varType = "boolean"
        elif self.TEMPORALS_STRING <= address < self.TEMPORALS_OBJECT:
            varType = "string"
        elif self.TEMPORALS_OBJECT <= address <= self.END_TEMPORALS:
            varType = "obj"
    else:
        self.error.definition(self.error.INVALID_MEMORY_ACCESS, stringAddress, None)

    memorySegment = [context, varType]
    return memorySegment
```

Arquitectura del Manejador de contextos



Dentro de todos los contextos la memoria temporal guarda únicamente variables sencillas (es decir, un entero, un string, etc.) Para el caso de los arreglos (guardados en memoria local y global) se necesitó hacer de instrucciones especiales, siendo estos los siguientes:

- ARRAY_DECLARE: Registra una nueva variable con el tamaño indicado. En el fondo esto sólo crea una lista del tamaño dado por el usuario.
- ARRAY_POS: Se utiliza cuando se debe acceder a un arreglo, guardando el índice brindado para poder hacer el corrimiento necesario a partir de la primer casilla de la lista.

En el programa se realiza un cambio de contexto cada vez que se ingresa a una rutina (por medio del GOSUB), asignando como contexto actual un nuevo contexto local. Este cambio de contexto se ve activado por el ERA, y cuando se detecta se guarda el contexto que se tenía previamente en una fila de contextos, para poder así regresar después. Al encontrarse cualquier RETURN se hace hace pop a la fila de contextos y se cambia el contexto actual por el que se tenía previamente.

Sin embargo, esto causaba complicaciones con el registro de parámetros, pues este ocurría después del ERA, pero para ese momento ya se había cambiado de contexto. Es decir, los parámetros estarían haciendo al nuevo contexto y no al contexto de donde vienen realmente. Esto se solucionó haciendo un cambio de contexto momentáneo al momento de registrar parámetros, cambiando al contexto anterior para buscar sus valores y cambiando al nuevo contexto para registrar dicho valor.

e). PRUEBAS DEL FUNCIONAMIENTO DEL LENGUAJE :

Fibonacci Recursivo

```
program fibRecursive;

function int fibo(var int fiboLimite, var int prueba){
    var int fiboLimiteAnt;
    var int fiboLimiteAntAnt;
    var int fiboAnt;
    var int fiboAntAnt;

    if(fiboLimite == 0)
    {
        return 0;
    }
    else
    {
        if(fiboLimite == 1){
            return 1;
        } else
        {
            fiboLimiteAnt = fiboLimite - 1;
            fiboLimiteAntAnt = fiboLimite - 2;

            fiboAnt = fibo(fiboLimiteAnt, prueba);
            fiboAntAnt = fibo(fiboLimiteAntAnt, prueba);

            return fiboAnt + fiboAntAnt;
        }
    }
}

main {
    var int resultado = 0;
    var int fiboLimite = 20;

    resultado = fibo(fiboLimite, 100);
    write(resultado);
}
```

```
C:\Users\dadel\Desktop\One-For-All>python one_for_all.py fibRecursive.txt

C:\Users\dadel\Desktop\One-For-All>python run.py program.ofa
1 Goto None None 28
2 == &11000 0 &27000
3 GotoF &27000 None 6
4 RETURN None None 0
5 Goto None None 28
6 == &11000 1 &27001
7 GotoF &27001 None 10
8 RETURN None None 1
9 Goto None None 28
10 - &11000 1 &23000
11 = &23000 None &11002
12 - &11000 2 &23001
13 = &23001 None &11003
14 ERA None None fibo
15 PARAM &11001 None &11001
16 PARAM &11002 None &11000
17 RETURN_ASSIGN None None &23002
18 GOSUB None None 2
19 = &23002 None &11004
20 ERA None None fibo
21 PARAM &11001 None &11001
22 PARAM &11003 None &11000
23 RETURN_ASSIGN None None &23003
24 GOSUB None None 2
25 = &23003 None &11005
26 + &11004 &11005 &23004
27 RETURN None None &23004
28 = 0 None &11000
29 = 20 None &11001
30 ERA None None fibo
31 PARAM 100 None &11001
32 PARAM &11001 None &11000
33 RETURN_ASSIGN None None &23000
34 GOSUB None None 2
35 = &23000 None &11000
36 WRITE None None &11000
37 END_WRITE None None None
```

Fibonacci Iterativo

```
program fibIterative;

var int fibo[100];

main {
    var int resultado = 0;
    var int fiboLimite = 20;
    var int counter = 2;
    var int fiboAnt = 0;
    var int fiboAntAnt = 0;

    fibo[0] = 0;
    fibo[1] = 1;

    if (fiboLimite == 0)
    {
        resultado = fibo[0];
    }

    if (fiboLimite == 1)
    {
        resultado = fibo[1];
    }

    while(counter <= fiboLimite)
    {
        fiboAntAnt = fibo[counter - 2];
        fiboAnt = fibo[counter - 1];

        fibo[counter] = fiboAntAnt + fiboAnt;

        counter = counter + 1;
    }

    resultado = fibo[fiboLimite];
    write(resultado);
}
```

```
C:\Users\dadel\Desktop\One-For-All>python one_for_all.py fibIterative.txt

C:\Users\dadel\Desktop\One-For-All>python run.py program.ofa
1 ARRAY_DECLARE 100 None &1000
2 Goto None None 3
3 = 0 None &11000
4 = 20 None &11001
5 = 2 None &11002
6 = 0 None &11003
7 = 0 None &11004
8 ARRAY_POS None None 0
9 = 0 None &1000
10 ARRAY_POS None None 1
11 = 1 None &1000
12 == &11001 0 &27000
13 GotoF &27000 None 16
14 ARRAY_POS None None 0
15 = &1000 None &11000
16 == &11001 1 &27001
17 GotoF &27001 None 20
18 ARRAY_POS None None 1
19 = &1000 None &11000
20 <= &11002 &11001 &27002
21 GotoF &27002 None 34
22 - &11002 2 &23000
23 ARRAY_POS None None &23000
24 = &1000 None &11004
25 - &11002 1 &23001
26 ARRAY_POS None None &23001
27 = &1000 None &11003
28 ARRAY_POS None None &11002
29 + &11004 &11003 &23002
30 = &23002 None &1000
31 + &11002 1 &23003
32 = &23003 None &11002
33 Goto None None 20
34 ARRAY_POS None None &11001
35 = &1000 None &11000
36 WRITE None None &11000
37 END_WRITE None None None
38 END None None None
6765
```

Factorial Recursivo

```
program factRecursive;

function int factorial(var int N, var int prueba)
{
    var int factAnt = 0;
    var int nAnt = 0;

    if(N == 1)
    {
        return 1;
    }
    else
    {
        nAnt = N - 1;
        factAnt = factorial(nAnt, prueba);
        return N * factAnt;
    }

}

main {
    var int resultado = 0;

    resultado = factorial(10, 100);
    write(resultado);
}
```

```
C:\Users\dadel\Desktop\One-For-All>python one_for_all.py factRecursive.txt

C:\Users\dadel\Desktop\One-For-All>python run.py program.ofa
1 Goto None None 18
2 = 0 None &11002
3 = 0 None &11003
4 == &11000 1 &27000
5 GotoF &27000 None 8
6 RETURN None None 1
7 Goto None None 18
8 - &11000 1 &23000
9 = &23000 None &11003
10 ERA None None factorial
11 PARAM &11001 None &11001
12 PARAM &11003 None &11000
13 RETURN_ASSIGN None None &23001
14 GOSUB None None 2
15 = &23001 None &11002
16 * &11000 &11002 &23002
17 RETURN None None &23002
18 = 0 None &11000
19 ERA None None factorial
20 PARAM 100 None &11001
21 PARAM 10 None &11000
22 RETURN_ASSIGN None None &23000
23 GOSUB None None 2
24 = &23000 None &11000
25 WRITE None None &11000
26 END_WRITE None None None
27 END None None None
3628800
```

Factorial Iterativo

```
program factIterative;

main {
    var int N = 10;
    var int resultado = 1;
    var int i = 1;

    while(i <= N)
    {
        resultado = (resultado * i);
        i = i + 1;
    }
    write(resultado);
}
```

```
C:\Users\dadel\Desktop\One-For-All>python one_for_all.py factRecursive.txt

C:\Users\dadel\Desktop\One-For-All>python run.py program.ofa
1 Goto None None 18
2 = 0 None &11002
3 = 0 None &11003
4 == &11000 1 &27000
5 GotoF &27000 None 8
6 RETURN None None 1
7 Goto None None 18
8 - &11000 1 &23000
9 = &23000 None &11003
10 ERA None None factorial
11 PARAM &11001 None &11001
12 PARAM &11003 None &11000
13 RETURN_ASSIGN None None &23001
14 GOSUB None None 2
15 = &23001 None &11002
16 * &11000 &11002 &23002
17 RETURN None None &23002
18 = 0 None &11000
19 ERA None None factorial
20 PARAM 100 None &11001
21 PARAM 10 None &11000
22 RETURN_ASSIGN None None &23000
23 GOSUB None None 2
24 = &23000 None &11000
25 WRITE None None &11000
26 END_WRITE None None None
27 END None None None
3628800
```

Bubble Sort

```
program bubbleSort;

var int arrayPrueba[10];

main {
    var int cant = 10;
    var int i = 0;

    var int swapped = 1;
    var int aux = 0;

    var int elemUno = 0;
    var int elemDos = 0;

    arrayPrueba[0] = 28;
    arrayPrueba[1] = 10;
    arrayPrueba[2] = 5;
    arrayPrueba[3] = 1;
    arrayPrueba[4] = 7;
    arrayPrueba[5] = 6;
    arrayPrueba[6] = 55;
    arrayPrueba[7] = 10;
    arrayPrueba[8] = 2;
    arrayPrueba[9] = 15;

    while(swapped == 1)
    {
        swapped = 0;
        i = 0;

        while(i < cant - 1)
        {
            if(arrayPrueba[i] < arrayPrueba[i + 1])
            {
                aux = arrayPrueba[i];
                arrayPrueba[i] = arrayPrueba[i + 1];
                arrayPrueba[i + 1] = aux;
                swapped = 1;
            }
            i = i + 1;
        }
    }

    i = 0;

    while(i < cant)
    {
        write(arrayPrueba[i]);
        i = i + 1;
    }
}
```

```
34 < &11001 &23000 &27001
35 GotoF &27001 None 54
36 ARRAY_POS None None &11001
37 + &11001 1 &23001
38 ARRAY_POS None None &23001
39 < &1000 &1000 &27002
40 GotoF &27002 None 51
41 ARRAY_POS None None &11001
42 = &1000 None &11003
43 ARRAY_POS None None &11001
44 + &11001 1 &23002
45 ARRAY_POS None None &23002
46 = &1000 None &1000
47 + &11001 1 &23003
48 ARRAY_POS None None &23003
49 = &11003 None &1000
50 = 1 None &11002
51 + &11001 1 &23004
52 = &23004 None &11001
53 Goto None None 33
54 Goto None None 29
55 = 0 None &11001
56 < &11001 &11000 &27003
57 GotoF &27003 None 64
58 ARRAY_POS None None &11001
59 WRITE None None &1000
60 END_WRITE None None None
61 + &11001 1 &23005
62 = &23005 None &11001
63 Goto None None 56
64 END None None None
1
2
5
6
7
10
10
15
28
55
```

Búsqueda en arreglo

```
program search;

var int arraySearch[10];

main {
    var int cant = 10;
    var int searchValue = 5;
    var int i = 0;

    arraySearch[0] = 28;
    arraySearch[1] = 10;
    arraySearch[2] = 5;
    arraySearch[3] = 1;
    arraySearch[4] = 7;
    arraySearch[5] = 6;
    arraySearch[6] = 55;
    arraySearch[7] = 10;
    arraySearch[8] = 2;
    arraySearch[9] = 15;

    while(arraySearch[i] != searchValue && i < cant)
    {
        i = i + 1;
    }

    if(i == cant)
    {
        write("Missing");
    }
    else
    {
        write("Found");
        write(i);
    }
}
```

```
6 ARRAY_POS None None 0
7 = 28 None &1000
8 ARRAY_POS None None 1
9 = 10 None &1000
10 ARRAY_POS None None 2
11 = 5 None &1000
12 ARRAY_POS None None 3
13 = 1 None &1000
14 ARRAY_POS None None 4
15 = 7 None &1000
16 ARRAY_POS None None 5
17 = 6 None &1000
18 ARRAY_POS None None 6
19 = 55 None &1000
20 ARRAY_POS None None 7
21 = 10 None &1000
22 ARRAY_POS None None 8
23 = 2 None &1000
24 ARRAY_POS None None 9
25 = 15 None &1000
26 ARRAY_POS None None &11002
27 != &1000 &11001 &27000
28 < &11002 &11000 &27001
29 && &27000 &27001 &27002
30 GotoF &27002 None 34
31 + &11002 1 &23000
32 = &23000 None &11002
33 Goto None None 26
34 == &11002 &11000 &27003
35 GotoF &27003 None 39
36 WRITE None None "Missing"
37 END_WRITE None None None
38 Goto None None 43
39 WRITE None None "Found"
40 END_WRITE None None None
41 WRITE None None &11002
42 END_WRITE None None None
43 END None None None
"Found"
2
```