**Tecnológico de Monterrey**

Diseño de compiladores

**Compilador de RedRobin**

Profesora del curso:

Profa. Elda Quiroga

Miembros del proyecto:

Omar Manjarrez Osornio A00815248

David Delgadillo Alvarez A01195995

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Omar Manjarrez

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

David Delgadillo

Fecha de entrega: 23 de Noviembre del 2016

Índice

[**Descripción del proyecto**](#_2r3t8tty6fay)[**4**](#_2r3t8tty6fay)

[Visión, objetivo y alcance de proyecto](#_hurmxjq6svt) [4](#_hurmxjq6svt)

[Visión](#_6f75n0jfozer) [4](#_6f75n0jfozer)

[Objetivo](#_m4e0zpaw34bm) [4](#_m4e0zpaw34bm)

[Alcance de proyecto](#_l0yl38hw8ioe) [4](#_l0yl38hw8ioe)

[Análisis de requerimientos y casos de uso generales](#_efhi7tnbcv5b) [4](#_efhi7tnbcv5b)

[Descripción de los principales test cases](#_394cuf7td5e) [6](#_394cuf7td5e)

[Descripción del proceso general seguido](#_pblfnfb9htpq) [6](#_pblfnfb9htpq)

[Bitácoras](#_ajt3b2sh9etr) [7](#_ajt3b2sh9etr)

[Reflexiones](#_gojyl3pfcfhh) [8](#_gojyl3pfcfhh)

[**Descripción del lenguaje**](#_df52h6gtntjm)[**9**](#_df52h6gtntjm)

[Nombre de lenguaje](#_wglwseyc2eco) [9](#_wglwseyc2eco)

[Descripción genérica de las características del lenguaje](#_vv8qwwpszq8t) [9](#_vv8qwwpszq8t)

[Descripción de los errores que pueden ocurrir](#_luy8ugg2c1pt) [9](#_luy8ugg2c1pt)

[**Descripción del lenguaje**](#_3xfho5lhtshv)[**10**](#_3xfho5lhtshv)

[Equipo de cómputo](#_w9iy3tu1jdoa) [10](#_w9iy3tu1jdoa)

[Descripción del análisis de léxico](#_afc7pi6ybsvq) [11](#_afc7pi6ybsvq)

[Tokens](#_1ij918z6rer4) [11](#_1ij918z6rer4)

[Palabras reservadas](#_81og0qpjcwj) [11](#_81og0qpjcwj)

[Descripción del análisis de sintaxis](#_1fuvux1ksppt) [12](#_1fuvux1ksppt)

[Descripción de generación de código intermedio](#_h4afgzgbbwli) [15](#_h4afgzgbbwli)

[Códigos de operación](#_fyp6atkj8ct9) [15](#_fyp6atkj8ct9)

[Direcciones virtuales](#_8n7mzhjhr7x3) [16](#_8n7mzhjhr7x3)

[Diagramas de sintaxis con acciones de semántica](#_sqst11nvqdc7) [17](#_sqst11nvqdc7)

[Tabla de consideraciones semánticas](#_t08iwn8oz1d1) [25](#_t08iwn8oz1d1)

[Descripción de la administración de memoria usado en la compilación](#_mug5bq2caxel) [26](#_mug5bq2caxel)

[**Descripción de la máquina virtual**](#_k66jqewh1q7b)[**30**](#_k66jqewh1q7b)

[Descripción del proceso de administración de memoria en ejecución](#_9lbrulr07rvz) [30](#_9lbrulr07rvz)

[Estructura de memoria global](#_354gtfiau6e0) [30](#_354gtfiau6e0)

[Estructura de memoria local](#_30aq987yemx9) [31](#_30aq987yemx9)

[Asociación entre las direcciones virtuales de compilación y las reales de ejecución](#_qlbynt4nituu) [32](#_qlbynt4nituu)

[**Pruebas del funcionamiento del lenguaje**](#_i842qm9ubd9y)[**33**](#_i842qm9ubd9y)

[Factorial cíclico](#_7xqfdqwtz9wz) [33](#_7xqfdqwtz9wz)

[Factorial recursivo](#_ziglw846qd92) [34](#_ziglw846qd92)

[Fibonacci cíclico](#_vl0kwvs0bzno) [36](#_vl0kwvs0bzno)

[Fibonacci recursivo](#_mm8h7h741bvy) [37](#_mm8h7h741bvy)

[Find en vector](#_xlyliyjmgox5) [39](#_xlyliyjmgox5)

[Sort de vector](#_uj9ajsa6yy1u) [41](#_uj9ajsa6yy1u)

[Pruebas propias](#_dvyf5iwm40uk) [46](#_dvyf5iwm40uk)

[**Listados documentados del proyecto**](#_iunvy8ikptls)[**52**](#_iunvy8ikptls)

[Módulos importantes de semántica](#_x2d1msprwzzc) [52](#_x2d1msprwzzc)

[Módulos importantes de máquina virtual](#_krpbxz88zc0g) [56](#_krpbxz88zc0g)

[Manual de usuario](#_nh658y77qff0) [59](#_nh658y77qff0)

# Descripción del proyecto

## Visión, objetivo y alcance de proyecto

### Visión

El propósito de nuestro proyecto es crear un lenguaje, y su respectivo compilador, orientado a objetos que facilite el aprendizaje de conceptos de este paradigma además de ofrecer una sintaxis simple para aumentar la productividad del programador.

### Objetivo

El objetivo del lenguaje es ser una herramienta de enseñanza del paradigma orientado a objetos utilizando una sintaxis simple, corta y sencilla de aprender.

### Alcance de proyecto

Crear un lenguaje con las características básicas de orientada a objetos teniendo un compilador que se pueda ejecutar desde terminal para hacer análisis léxico y semántico sobre un código fuente con la sintaxis de nuestro lenguaje RedRobin que genera un archivo de formato “txt” representando el código objeto. Se desarrolla también una máquina virtual que se pueda ejecutar desde terminal también sobre uno de los códigos objetos generados por el compilador para realizar la ejecución del programa realizado por el usuario.

El lenguaje RedRobin soporta las operaciones básicas de expresiones aritméticas (suma, resta, multiplicación, división) y lógicas (mayor o igual que, menor o igual que, igual que, diferente de). Aunado a esto, se soportan las generaciones de ciclos, condicionales e invocación a funciones.

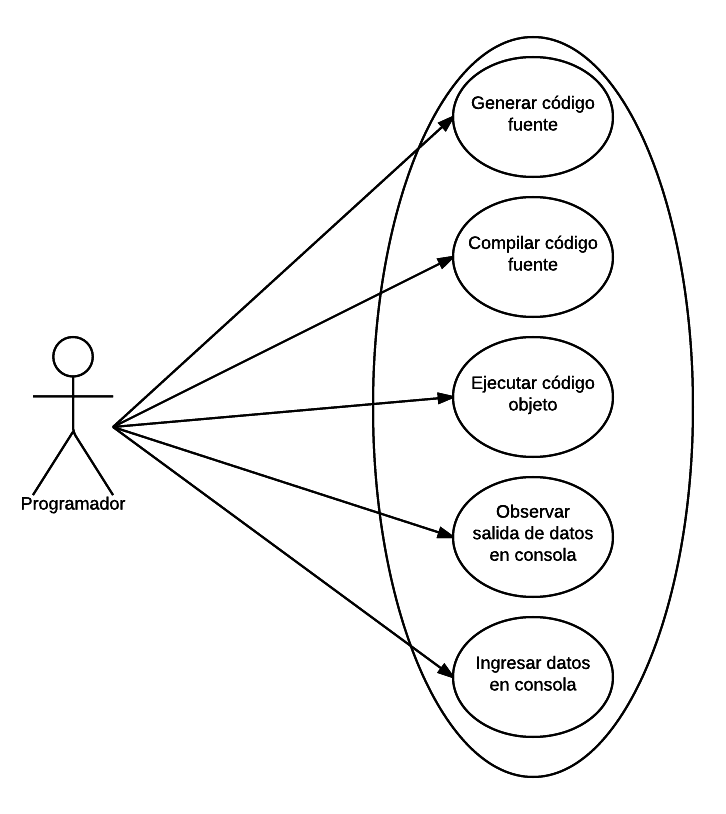
La característica de mayor relevancia es el manejo de clases y objetos permitiendo definir sus características (atributos de la clase), definir posibles comportamientos del objeto (métodos de la clase) y soportar la herencia de un nivel y composición de uno sólo también. Todo esto permite envolver atributos de distintas características o variables en un tipo de estructura única.

## Análisis de requerimientos y casos de uso generales

Se enlistan en lo siguiente la funcionalidad que se implementa para la compilación y ejecución del lenguaje RedRobin como requerimientos del proyecto:

* Validación de léxico del código fuente en lectura
* Validación de sintaxis del código fuente en lectura
* Validación de semántica del código fuente en lectura
* Manejo de tabla de variables
* Generación de cuádruplos
* Generación de archivo con código objeto
* Lectura y ejecución de código objeto con máquina virtual
* Manejo de memoria de ejecución
* Asignación de valores a variables
* Implementación de expresiones aritméticas
* Implementación de operaciones relacionales
* Implementación de ciclos
* Implementación de condicionales
* Implementación de datos de entrada
* Implementación de datos de salida a pantalla
* Implementación de invocación de funciones
* Implementación de definición e implementación de clases
* Implementación de herencia en un nivel
* Implementación de composición en un nivel
* Impresión en terminal de errores de compilación o ejecución

Se muestra en lo siguiente el diagrama de casos de uso general para el programador:



## Descripción de los principales test cases

La compilación de RedRobin y la máquina virtual para ejecutarla arrojan ciertos tipos de errores que se imprimen en terminal:

* Error léxico o sintaxis: Al realizar el “parsing” del código fuente se arroja mensaje de error de sintaxis o por atrapar un token no definido.
* Error de semántica: se arroja mensaje en terminal cuando hubo una incongruencia para la compilación del programa como tipos incompatibles por poner un ejemplo. Se enlista mas a detalle:
  + Error de “type mismatch”: como se mencionaba, cuando se trataba de realizar operaciones sobre tipo de valores incongruentes.
  + Error de declaración repetitiva: se arroja cuando se quiere declarar una variable o nombre de función que ya fue declarada dentro del mismo alcance o entorno global o local.
  + Error de firma: se arroja cuando la invocación de una función no coinciden sus argumentos con la firma del método definido.
* Error de ejecución: se arrojan al momento de andar ejecutando los cuádruplos obtenidos del código objeto. Se describen los distintos en lo siguiente:
  + Error de fuera de límite: se arroja cuando se trata de ingresar a una celda inexistente dentro de la estructura del arreglo por estar fuera de los límites de tamaño. Es el error típico de offset.
  + Error de falta de valor de retorno: se arroja cuando al invocar una función con variable de retorno termina su procedimiento pero no realizó ningún retorno de valor.

## Descripción del proceso general seguido

Se siguieron las mismas entregas propuestas por la profesora del curso para ir realizando el compilador y la máquin virtual. En términos generales empezar con la generación de la gramática regular para nuestro lenguaje, generar el archivo con la semántica en compilación y a la par cuando ya estaba muy avanzado empezar a desarrollar la máquina virtual.

La organización entre los miembros del equipo fue que David Delgadillo fue el encargado principal de general las reglas de la gramática y jugar con las ambigüedades que se presentaban para minimizarlas poco a poco y finalmente erradicarlas por completo. Omar Manjarrez fue el encargado principal de generar las funciones y reglas para la semántica diseñando la estructura de la tabla de variables, del manejo de direcciones de memoria y la generación de la mayoría de los cuádruplos. En cuanto a la generación del código objeto y la ejecución de la misma con la máquina virtual fue responsabilidad de David Delgadillo.

### Bitácoras

Semana 1 (Septiembre 19-23) Léxico y sintaxis:

Se diseñó los diagramas de sintaxis para el lenguaje que se le puso el nombre de RedRobin. Se discutió si se realizaba un lenguaje en español o en inglés pero se inclinó por el lenguaje anglosajón. Se inició con las reglas de gramática para el scanner pero no se completó. Lo que sí se completó fue el lexer incluyendo todas las palabras reservadas y tokens que manejaría el lenguaje.

Semana 2 (Octubre 3-7) Tablas de variables y Directorio de Procedimientos. Semántica básica de variables:

Se terminó de realizar las reglas de la gramática pero con ambigüedades de reduce/reduce y varias más de shift/reduce. Con esto había código fuente que si se compilaba o que no marcara error de sintaxis pero en otras ocasiones si marcaba cuando no debía. En cuanto a la semántica, se alcanzó a realizar que se la identificación de las variables declaradas repetidas o no sin estar sujetos al alcance local o global.

Semana 3 (Octubre 10-14) Cubo semántico y Generación de código de expresiones aritméticas:

Con ciertas adecuaciones y cambios a la reglas de gramática del lenguaje, se erradicaron todas las ambigüedades de reduce/reduce por lo que ya se parsea el código de manera correcta. Pero el haber eliminado las ambigüedades de r/r generó muchas más de s/r. Por otro lado se implementó el directorio de procedimientos, se validó la declaración múltiple de variables ahora sí ajustadas a su mismo alcance local y global y se hizo la estructura con el cubo semántico. No se alcanzó a empezar la generación de código.

Semana 4 (Octubre 17-21) Generación de código de estatutos secuenciales y estatutos condicionales:

Se hizo una refactorización a las reglas de gramática para manejar las expresiones y eso significó eliminar todas las ambigüedades que existían en la gramática. Se realizó la generación de cuádruplos para expresiones aritméticas.

Semana 5 (Septiembre 24-28) Generación de código de funciones y mapa de memoria de ejecución para la máquina virtual:

Se diseñó los diagramas de sintaxis para el lenguaje que se le puso el nombre deSe genera código de operaciones relacionales y ciclos while. Se diseñó y desarrolló el mapa de memoria.

Semana 6 (Noviembre 31-4) Máquina virtual para ejecutar expresiones aritméticas y estatutos secuenciales:

Se hizo la implementación de la generación de cuádruplos de ciclos for, de las condicionales y se empezó a trabajar en la estructura de memoria para las clases y objetos.

Semana 7 (Noviembre 7-11) Generación de código de arreglos y máquina virtual para ejecución de condicionales:

La generación de código de funciones está casi terminada excepto retornos de valor. Se desarrolla la ejecución de expresiones aritméticas en máquina virtual y se sigue preparando la semántica para el manejo de código orientado a objetos.

Semana 8 (Noviembre 14-18) 1era versión de documentación y máquina virtual ejecutando una parte de la aplicación:

La generación de código de arreglos está terminado. Se hace la ejecución de condicionales en máquina virtual y ya se realiza la generación de código para retorno de valores en función y referencias.

Semana final (Noviembre 19-22) Entrega de proyecto

Se terminó de realizar toda la máquina virtual y la compilación y generación de código relacionado para orientada a objetos. Se realiza toda la documentación en dos días.

### Reflexiones

David Delgadillo

La realización de este trabajo te da la oportunidad de tomar conciencia y obtener una mucho mejor comprensión de lo que está pasando cada vez que andamos compilando y ejecutando cualquier concepto que sea relacionado a indicar instrucciones de lógica para la realización de algo. Esta fue sin duda uno de los proyectos más integrales por tener que aplicar estructuras de datos, orientada a objetos, análisis y diseño de algoritmos además de las etapas de ciclo de vida del software.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Omar Manjarrez

El curso de compiladores y su proyecto fue indudablemente una de las materias de mayor reto y curiosidad. El realizar un lenguaje orientada a objetos me exprimió en mis habilidades para la creatividad y competencia de análisis por todo el diseño e implementación que se tuvo que ir planeando para cumplir con la implementación del producto. Desarrollarlo fue muy agotador pero a la vez muy divertido pues generaba mucho sentimiento de superación.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Descripción del lenguaje

## Nombre de lenguaje

RedRobin

## Descripción genérica de las características del lenguaje

RedRobin es un lenguaje de programación que soporta las características básica además de expresiones aritméticas y relaciones, condicionales, estatutos de ciclos funciones predefinidas de salida a pantalla, lectura de usuario mediante el teclado y casteo de valores válidos a strings, enteros y decimales. A esto se le suma el manejo de arreglos para tipo de datos primitivos.

También se soporta las características básicas del paradigma de orientada a objetos incluidos manejo de atributo y métodos públicos y privados, herencia en un nivel, y composición y herencia.

## Descripción de los errores que pueden ocurrir

* Error léxico o sintaxis: Al realizar el “parsing” del código fuente se arroja mensaje de error por obtener un elemento que no pertenece al vocabulario de tokens.
* Error sintaxis: Al ir realizando un parsing del código fuente, se identifica obtiene un token no esperado.
* Error de semántica: se arroja mensaje en terminal cuando hubo una incongruencia para la compilación del programa como tipos incompatibles por poner un ejemplo. Se enlista mas a detalle:
  + Error de “type mismatch”: como se mencionaba, cuando se trataba de realizar operaciones sobre tipo de valores incongruentes.
  + Error de declaración repetitiva: se arroja cuando se quiere declarar una variable o nombre de función que ya fue declarada dentro del mismo alcance o entorno global o local.
  + Error de firma: se arroja cuando la invocación de una función no coinciden sus argumentos con la firma del método definido.
* Error de ejecución: se arrojan al momento de andar ejecutando los cuádruplos obtenidos del código objeto. Se describen los distintos en lo siguiente:
  + Error de fuera de límite: se arroja cuando se trata de ingresar a una celda inexistente dentro de la estructura del arreglo por estar fuera de los límites de tamaño. Es el error típico de offset.
  + Error de falta de valor de retorno: se arroja cuando al invocar una función con variable de retorno termina su procedimiento pero no realizó ningún retorno de valor.
  + Error de división entre 0: se trata de realizar una operación aritmética donde el denominador tiene un valor de 0.

# Descripción del lenguaje

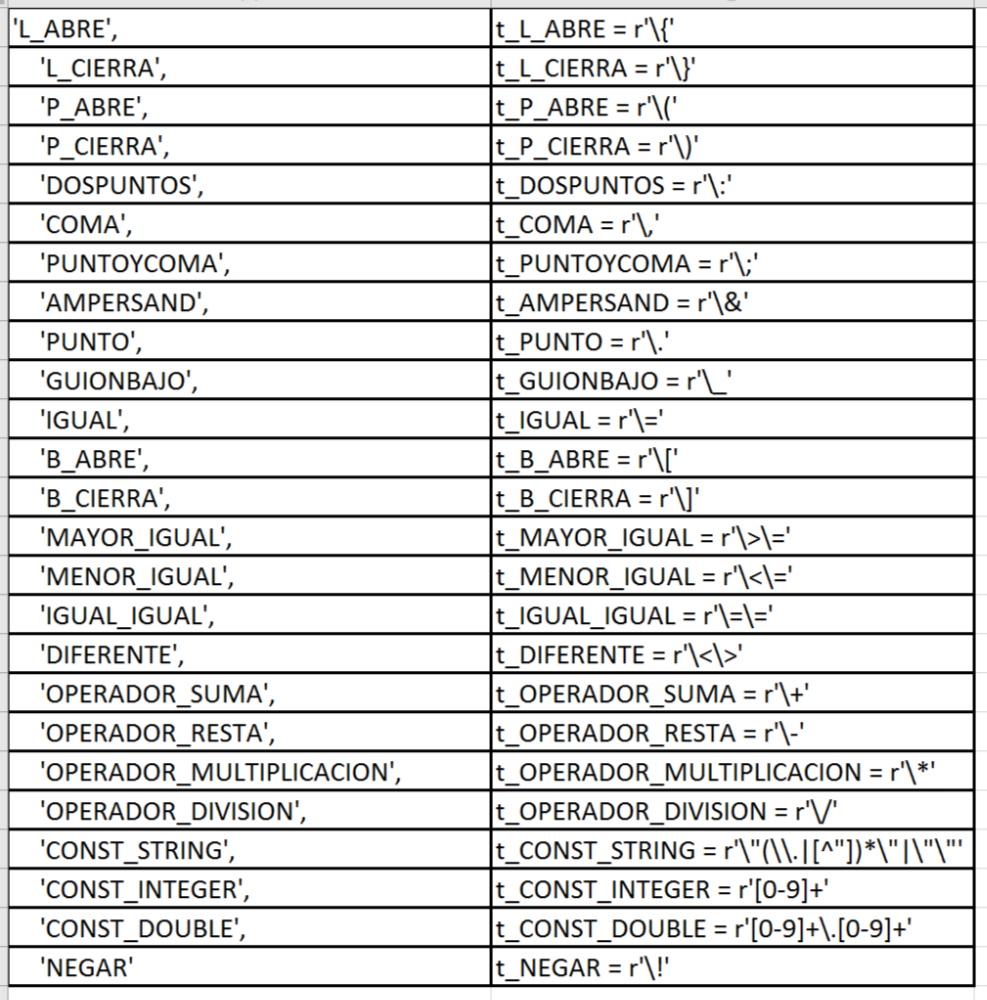
## Equipo de cómputo

El compilador está desarrollado en Python 3.5.2. Desarrollado de manera simultánea en un sistema operativo Windows 10 y un sistema operativo Mac OS X.

Es necesario bajar las librerías de python Numpy y PLY para el funcionamiento del compilador y la máquina virtual.

## Descripción del análisis de léxico

### Tokens



### Palabras reservadas

'class' : 'CLASS',

'RedRobin' : 'REDROBIN',

'public' : 'PUBLIC',

'empty' : 'EMPTY',

'secret' : 'SECRET',

'number' : 'NUMBER',

'real' : 'REAL',

'string' : 'STRING',

'bool' : 'BOOL',

'inherit' : 'INHERIT',

'give' : 'GIVE',

'for' : 'FOR',

'in' : 'IN',

'step' : 'STEP',

'while' : 'WHILE',

'if' : 'IF',

'elif' : 'ELIF',

'else' : 'ELSE',

'and' : 'AND',

'or' : 'OR',

'func' : 'FUNCTION',

'true' : 'TRUE',

'false' : 'FALSE',

'print' : 'PRINT',

'read' : 'READ',

'toNumber' : 'TONUMBER',

'toReal' : 'TOREAL',

'toString' : 'TOSTRING'

### 

## Descripción del análisis de sintaxis

Reglas de la gramática de RedRobin:

\*\*\*E: significa vacío

program --> CLASS REDROBIN L\_ABRE cuerpoprogram L\_CIERRA

cuerpoprogram --> codigo REDROBIN P\_ABRE P\_CIERRA L\_ABRE cuerpofuncion L\_CIERRA

codigo --> clases codigo

| funciones codigo

| E

funciones --> FUNCTION privilages valor\_retorno ID P\_ABRE parametros P\_CIERRA L\_ABRE cuerpofuncion L\_CIERRA

valor\_retorno --> tipovariable

| EMPTY

privilages --> SECRET

| PUBLIC

tipovariable --> NUMBER

| REAL

| STRING

| BOOL

| ID

parametros --> tipovariable posiblesbrackets DOSPUNTOS ID mas\_ids mas\_parametros

| E

mas\_ids --> COMA ID mas\_ids

| E

mas\_parametros --> PUNTOYCOMA parametros

| E

posiblesbrackets --> B\_ABRE B\_CIERRA

| E

clases --> CLASS ID herencia L\_ABRE cuerpoclase L\_CIERRA

herencia --> INHERIT ID

| empty

cuerpoclase --> privilages declaracion mas\_cuerpoclase

| funciones mas\_cuerpoclase

mas\_cuerpoclase --> cuerpoclase

| E

cuerpofuncion --> declaracion cuerpofuncion

| asignacion cuerpofuncion

| condicional cuerpofuncion

| ciclo cuerpofuncion

| funcionsinretorno PUNTOYCOMA cuerpofuncion

| io PUNTOYCOMA cuerpofuncion

| retorno cuerpofuncion

| E

retorno --> GIVE expresion PUNTOYCOMA

funcionsinretorno --> ID composicion\_atributo P\_ABRE argumentos P\_CIERRA

argumentos --> valorargumentos mas\_argumentos

| E

mas\_argumentos --> COMA valorargumentos mas\_argumentos

| E

valorargumentos --> AMPERSAND identificador

| expresion

composicion\_atributo --> PUNTO ID

| PUNTO ID PUNTO ID

| E

io --> PRINT P\_ABRE argumentosPrint P\_CIERRA

| READ P\_ABRE identificador P\_CIERRA

argumentosPrint --> argumentoPosible mas\_prints

| E

mas\_prints --> COMA argumentoPosible mas\_prints

| E

argumentoPosible --> expresion

ciclo --> cicloestatico

| ciclodinamico

cicloestatico --> FOR ID IN P\_ABRE expresion GUIONBAJO expresion P\_CIERRA STEP valor L\_ABRE cuerpofuncion L\_CIERRA

ciclodinamico --> WHILE P\_ABRE expresion P\_CIERRA L\_ABRE cuerpofuncion L\_CIERRA

condicional --> IF P\_ABRE expresion P\_CIERRA L\_ABRE cuerpofuncion L\_CIERRA condiciones\_elif condicion\_else

condiciones\_elif --> ELIF P\_ABRE expresion P\_CIERRA L\_ABRE cuerpofuncion L\_CIERRA condiciones\_elif

| E

condicion\_else --> ELSE L\_ABRE cuerpofuncion L\_CIERRA

| E

asignacion --> identificador IGUAL expresion PUNTOYCOMA

declaracion --> tipovariable ID declara\_arreglo\_o\_iniciacion mas\_declaraciones PUNTOYCOMA

declara\_arreglo\_o\_iniciacion --> B\_ABRE CONST\_INTEGER B\_CIERRA

| IGUAL expresion

| empty

mas\_declaraciones --> COMA ID declara\_arreglo\_o\_iniciacion mas\_declaraciones

| E

expresion --> expresionii mas\_expresion

mas\_expresion --> OR expresion

| E

expresionii --> expresioniii mas\_expresionii

mas\_expresionii --> AND expresionii

| E

expresioniii --> expresioniv mas\_expresioniii

mas\_expresioniii --> operadorrelacional expresioniii

| E

expresioniv --> expresionv mas\_expresioniv

mas\_expresioniv --> operadortermino expresioniv

| E

expresionv --> expresionvi mas\_expresionv

mas\_expresionv --> operadorfactor expresionv

| E

expresionvi --> valor

| negacion P\_ABRE expresion P\_CIERRA

negacion --> NEGAR

| OPERADOR\_RESTA

| E

operadorrelacional --> MAYOR\_IGUAL

| MENOR\_IGUAL

| IGUAL\_IGUAL

| DIFERENTE

operadortermino --> OPERADOR\_SUMA

| OPERADOR\_RESTA

operadorfactor --> OPERADOR\_MULTIPLICACION

| OPERADOR\_DIVISION

valor --> negativo valorAdapter

valorAdapter --> identificador

| invocacion

| CONST\_STRING

| valorbooleano

| CONST\_INTEGER

| CONST\_DOUBLE

invocacion --> ID composicion\_atributo P\_ABRE argumentos P\_CIERRA

| TONUMBER P\_ABRE argumentoPosible P\_CIERRA

| TOREAL P\_ABRE argumentoPosible P\_CIERRA

| TOSTRING P\_ABRE argumentoPosible P\_CIERRA

valorbooleano --> TRUE

| FALSE

negativo --> OPERADOR\_RESTA

| E

identificador --> ID composicion\_atributo arreglo

arreglo --> B\_ABRE expresion B\_CIERRA

| E

## Descripción de generación de código intermedio

### Códigos de operación

Número de operación : Identificador de función en Máquina Virtual, #Operación

12 : suma, #+

13 : resta, #-

14 : multiplica, #\*

15 : divide, #/

16 : asigna, #=

17 : mayoroigual, #>=

18 : menoroigual, #<=

19 : esigual, #==

20 : esdiferente, #<>

21 : condicionalor, #or

22 : condicionaland, #and

50 : goTo, #goto

51 : goTof, #gotof

53 : era, #era - Genera alcance de la nueva memoria local para la función que se invoca

54 : irASubrutina, #gosub

55 : parametro, #param

56 : imprimir, #print

57 : lecturaTeclado, #read - Pide al usuario la entrada de un dato y almacena a una variable

58 : convierteANumero, #toNumber - Casteo de un valor a entero

59 : convierteAReal, #toReal - Casteo de un valor a decimal

60 : convierteAString, #toString - Casteo de un valor a string

62 : give, #give - Comando de retorno de un valor en una función

63 : terminaProcedimiento, #endproc

64 : terminaPrograma, #endprogram

65 : referencia, #ref - Actualiza valores de variables enviados como referencia a función

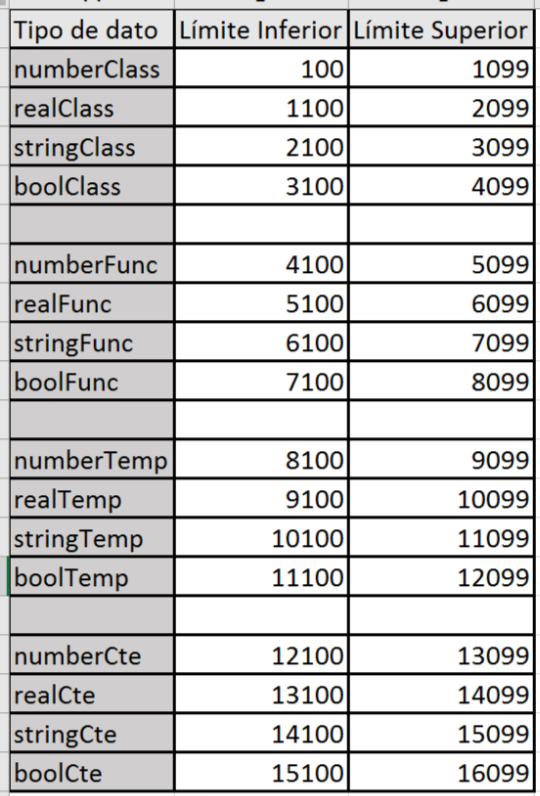
66 : verificarLimites, #ver - Valida que una indexación esté dentro de los límites

67 : asignaRetorno, #retu - Asigna a variable temporal el valor de retorno de una función

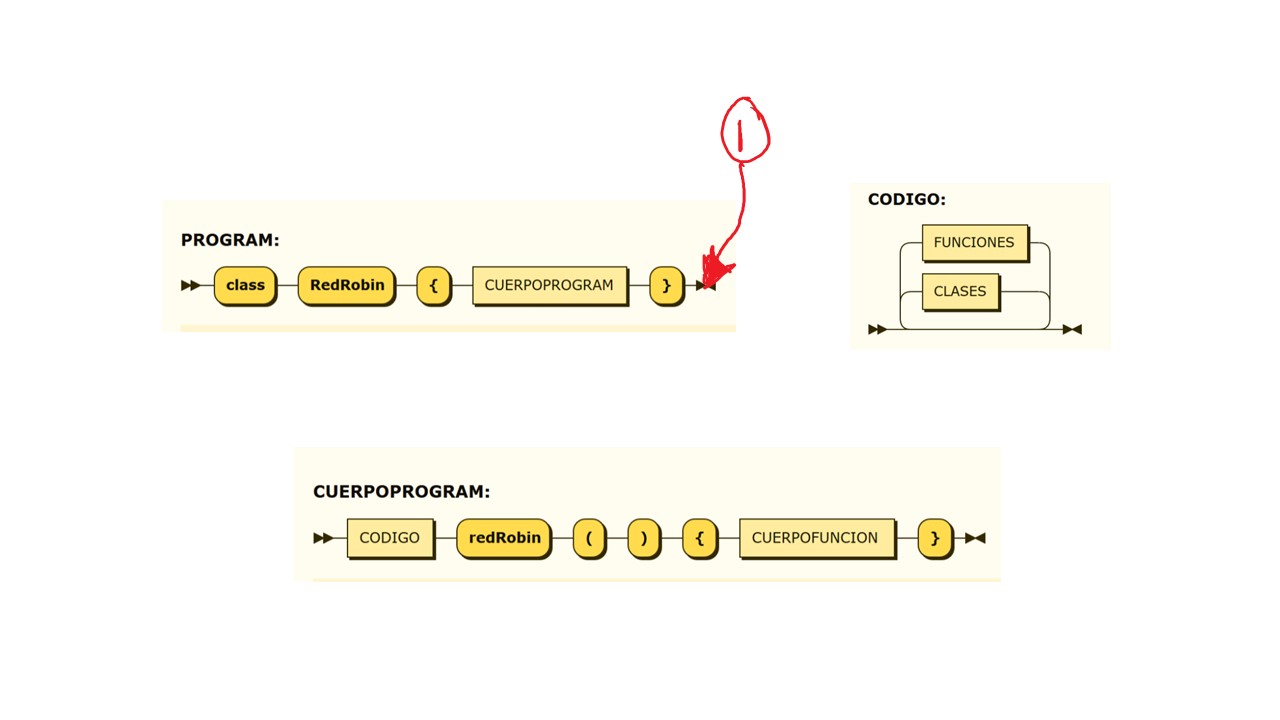
68 : negarBoleano, #not - Niega un valor booleano

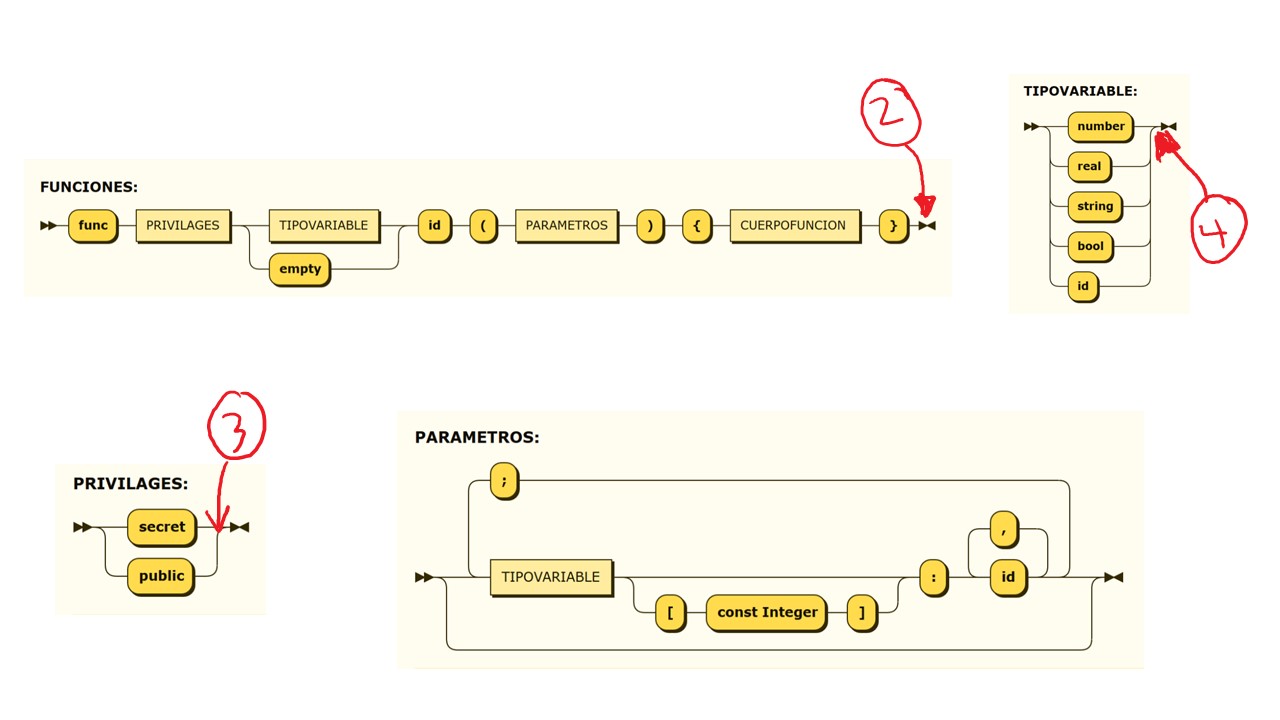
69 : asignaReferencia #setref - Inicializa los valores en el nuevo alcance local de una función según los valores que tienen los del alcance antiguo

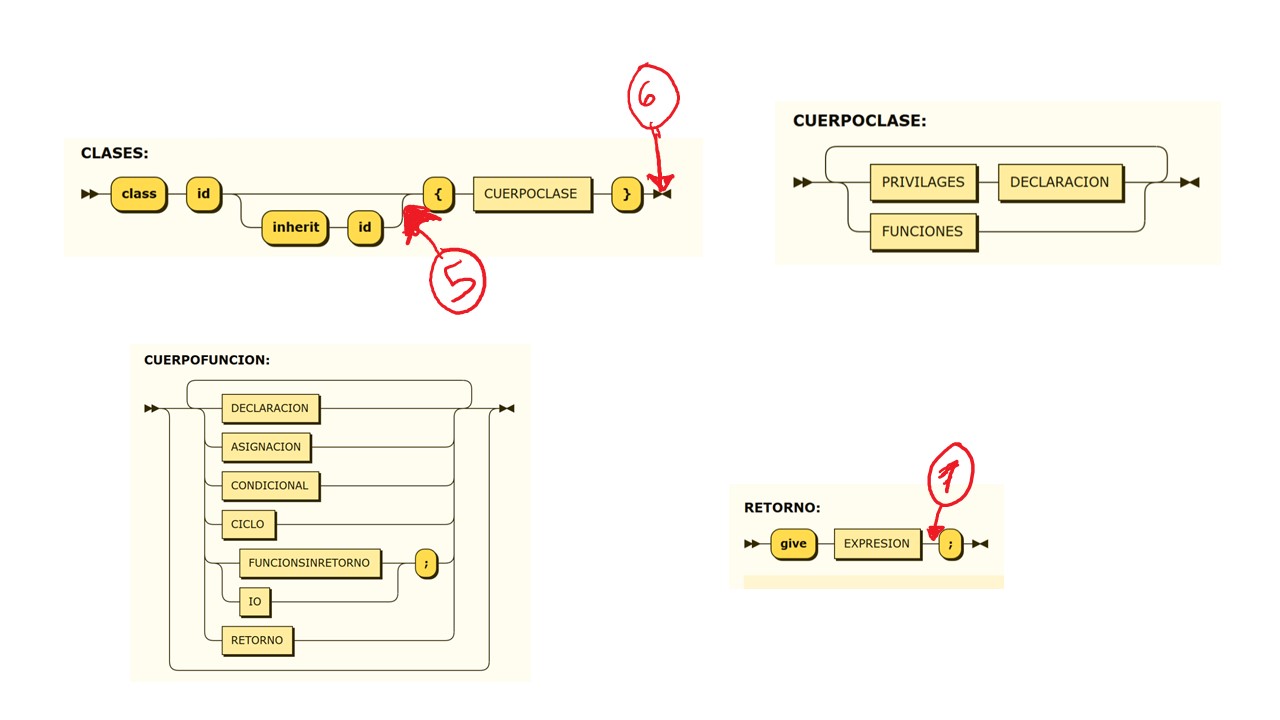
### Direcciones virtuales

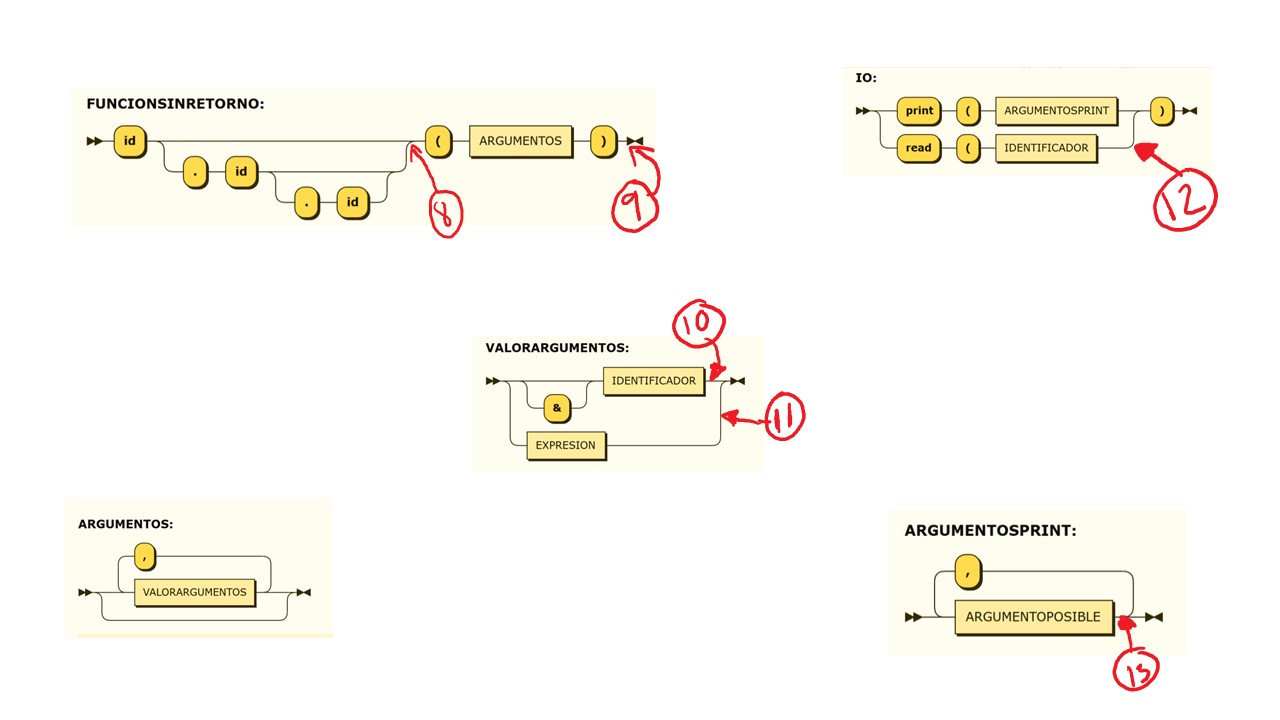


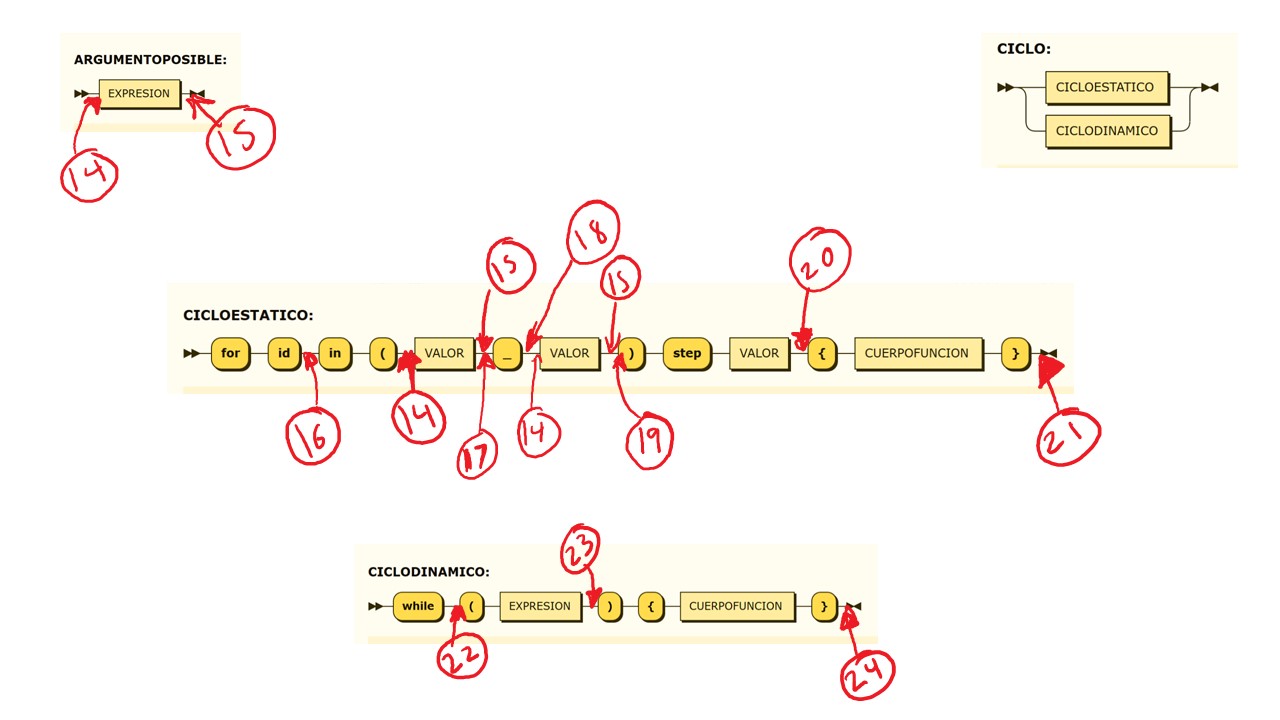
### Diagramas de sintaxis con acciones de semántica

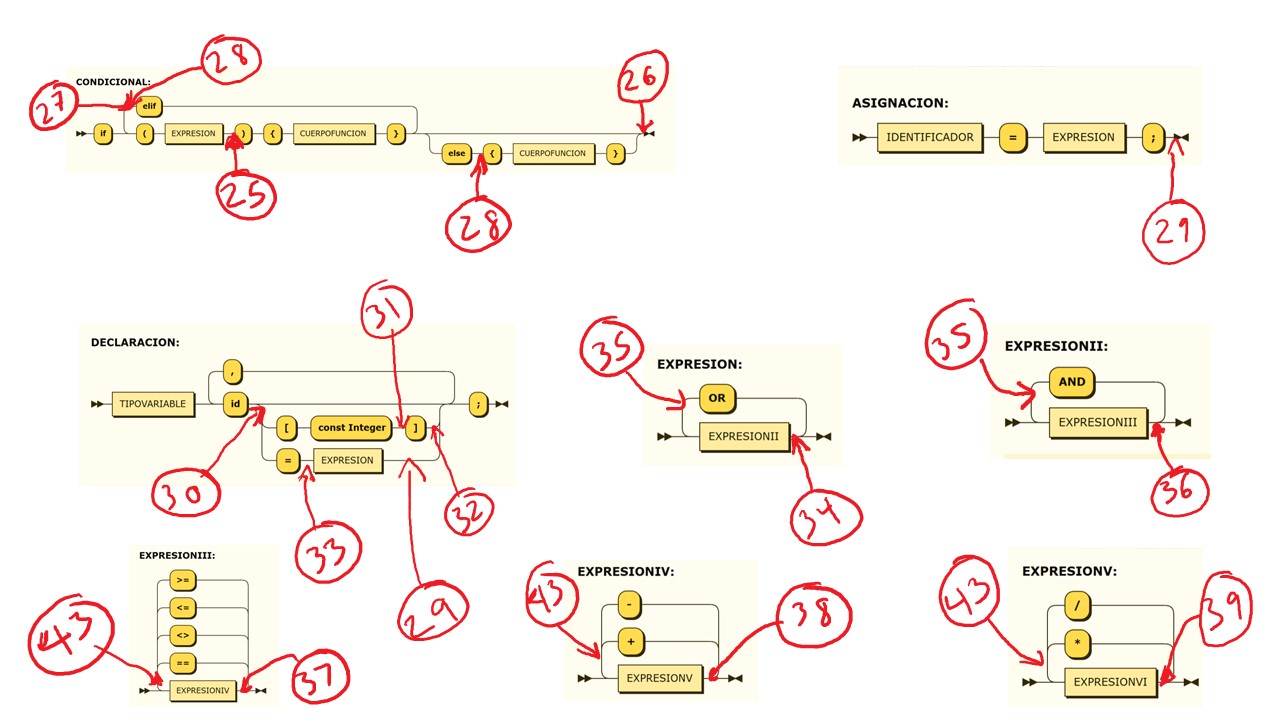


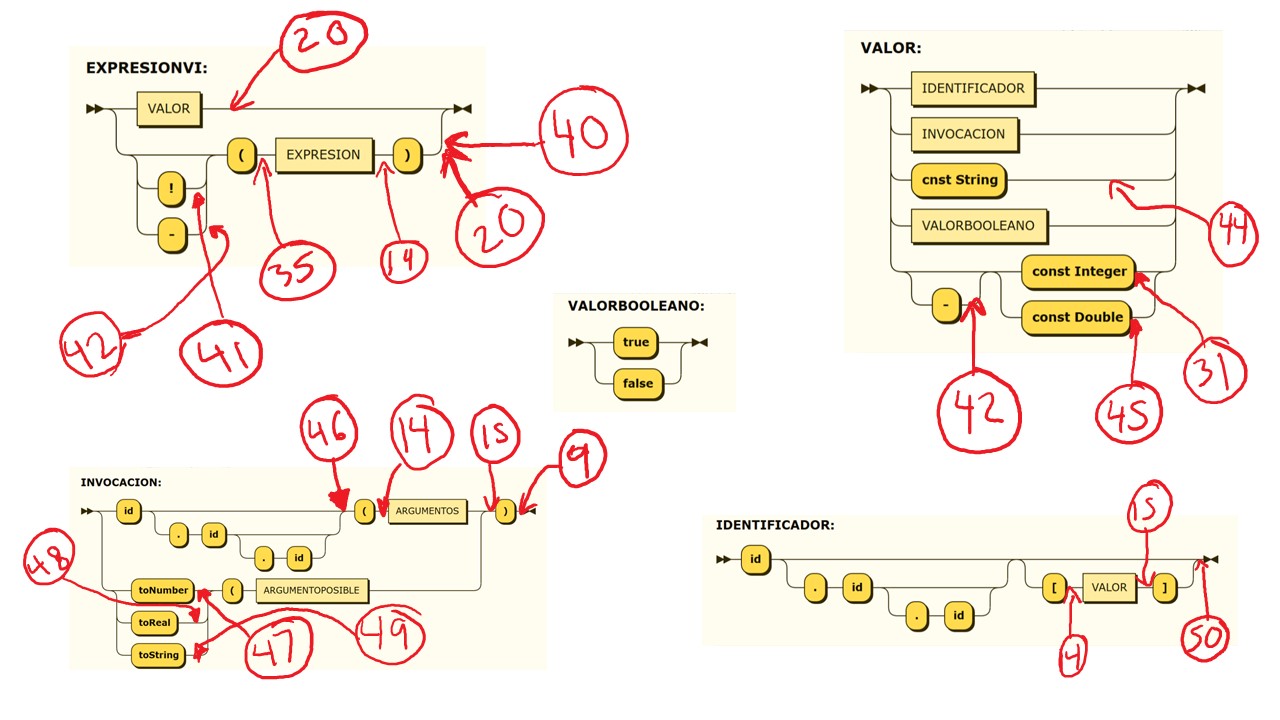








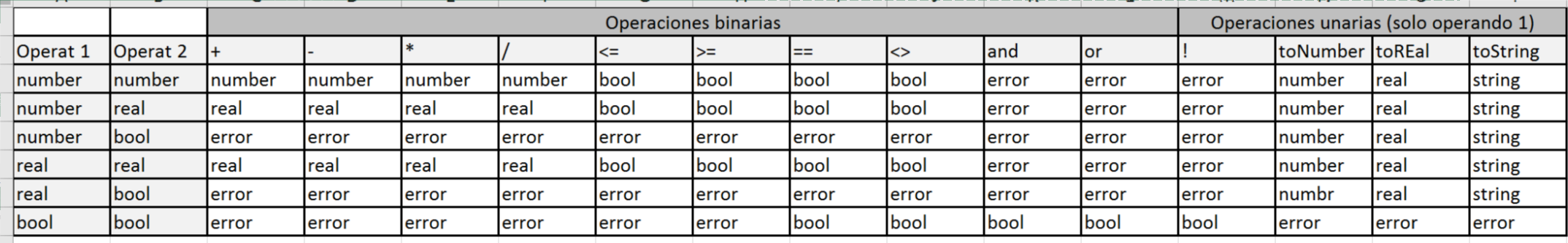




Descripción de acciones de semántica numeradas:

1. Crear cuádruplo endprogram
2. Crear cuádruplo endproc y poner scope de función vacio.
3. Guardar último privilegio usado
4. Guardar último tipo de dato usado
5. Crea una nueva clase añadiendo atributos de posible padre.
6. Poner scope de class de vuelta a RedRobin
7. Crear cuádruplo [give, dirRetornada, -1, varGlobalRetorno] y cuádruplo [endproc]
8. Guardar en hashRef los atributos de instancia que pudieran ser modificados por la función sin retorno
9. Genera quadruplos setref, ref, y gosub. Agrega el retorno de la función (en caso de haber) a la pila de operandos.
10. Agregar a hashRef el argumento enviado como referencia
11. Genera cuádruplo param con la dirección de memoria de la expresión enviada como parámetro.
12. Crear cuádruplo read con la dirección tope de operandos
13. Crear cuádruplo print con la dirección tope de operandos
14. Añadir fondo falso a la pila de operadores
15. Remover fondo falso de la pila de operadores
16. Válida variable de control del ciclo for y la inserta a la pila de operandos
17. Crear cuádruplo para almacenar el start del for en la variable de control
18. Añade a la pila de saltos el número de cuádruplos
19. Generar cuádruplo que verifica si la variable de control sigue siendo menor al límite superior.
20. Revisar si hay negativos pendientes en la pila de operadores. Resolverlos si es el caso
21. Crear cuádruplo para agregar el valor del step a la variable de control y otro para goto que va a la condición del ciclo.
22. Agregar la cantidad a de cuádruplos a la pila de saltos.
23. Crear gotof según la condicional y Agregar la cantidad de cuádruplos - 1 a la pila de saltos.
24. Crear cuádruplo goto a la dirección de retorno. Rellenar cuádruplo de gotof con la cantidad de cuádruplos.
25. Crear cuádruplo gotof con la condición y agregar la cantidad de cuádruplos a la pila de saltos
26. Rellenar todos los cuádruplos gotof pendientes con la cantidad de cuádruplos. Igualar cantidad de elif pendientes a 0
27. Aumentar la cantidad de elif pendientes en 1
28. Crear cuádruplo goto, añadir a la pila de saltos la cantidad de cuadruplos -1 y rellenar el tope de la pila de saltos con la cantidad de cuádruplos.
29. Crear cuádruplo de asignación con los 2 últimos valores de la pila de operandos.
30. Da de alta la variable en el directorio de procedimientos
31. Crea la dirección de memoria que almacenará la constante entera usada. Añadirla a la pila de operandos.
32. Añade en el directorio de procedimientos el tamaño a un identificador ya declarado.
33. Añade a la pila de operandos la variable previamente declarada
34. Revisar si hay ors pendientes en la pila de operadores. Resolverlos si es el caso
35. Añade a la pila de operadores el operador utilizado
36. Revisar si hay ands pendientes en la pila de operadores. Resolverlos si es el caso
37. Revisar si hay operadores relacionales pendientes en la pila de operadores. Resolverlos si es el caso.
38. Revisar si hay sumas pendientes en la pila de operadores. Resolverlas si es el caso
39. Revisar si hay multiplicaciones pendientes en la pila de operadores. Resolverlas si es el caso
40. Revisar si hay negaciones pendientes en la pila de operadores. Resolverlas si es el caso.
41. Añadir el operador not a la pila de operadores.
42. Añadir el operador negativo a la pila de operadores
43. Añadir operador a la pila de operadores
44. Crea la dirección de memoria que almacenará la constante string usada. Añadirla a la pila de operandos.
45. Crea la dirección de memoria que almacenará la constante double usada. Añadirla a la pila de operandos
46. Limpiar hashes auxiliar para el manejo de parámetros. Inicializar contador de parametros. Enviar por referencia variables necesarias
47. Genera el cuádruplo de operación de casteo a número
48. Genera el cuádruplo de operación de casteo a real
49. Genera el cuádruplo de operación de casteo a string
50. Validar el identificador y agregarlo a la pila de operandos.

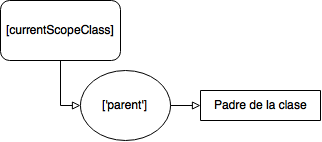
### Tabla de consideraciones semánticas



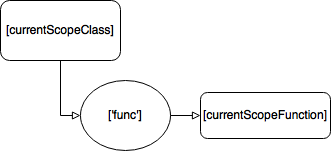
## Descripción de la administración de memoria usado en la compilación

* 1. dirProced

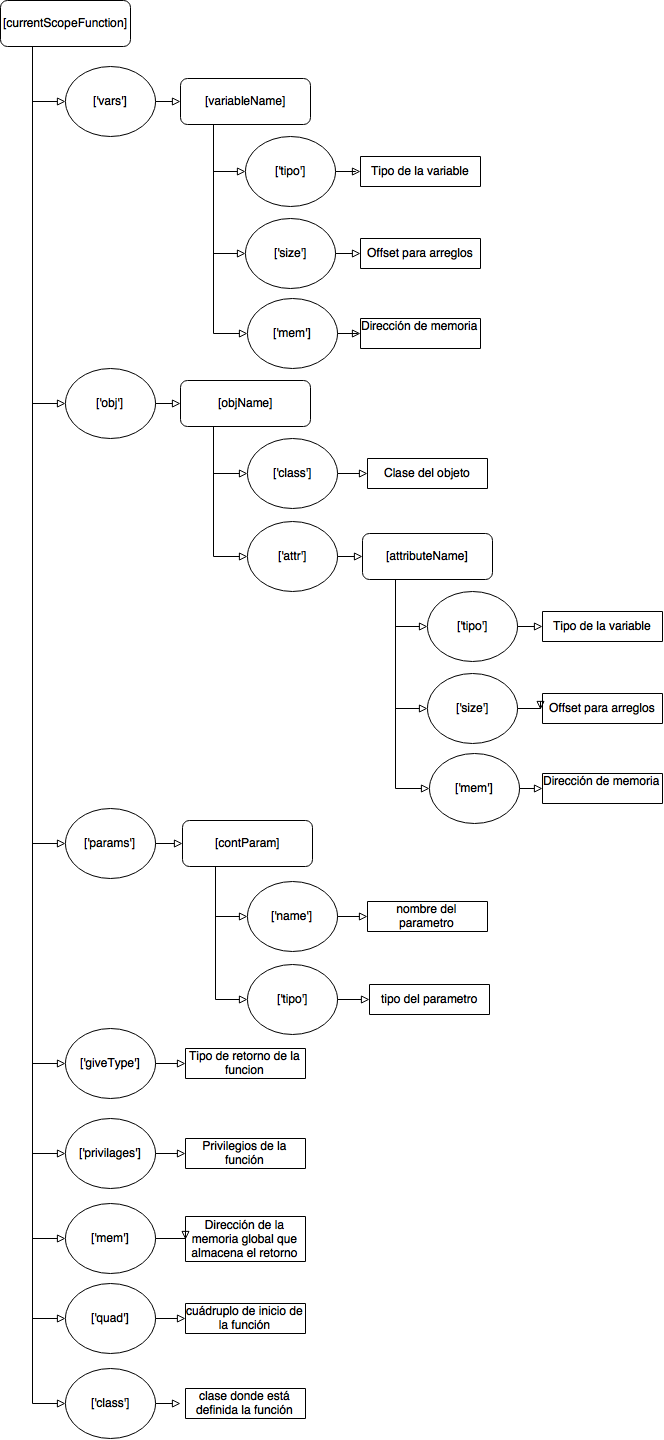
Durante compilación, dirProced es la estructura de datos principal que se encarga de almacenar todos los elementos del lenguaje en su correspondiente Scope. Esta estructura está conformado por un diccionario de diccionario de múltiples niveles, que va creciendo según se necesite. El primer nivel contiene contiene la llave currentScopeClass, a partir de la cual se puede ir navegando por la estructura como se muestra en los siguientes diagramas.



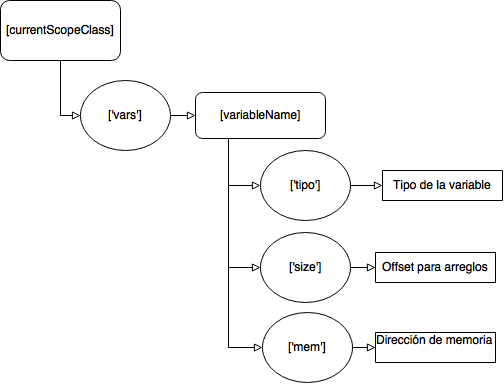
Con la llave ‘parent’ accedemos al padre de la clase currentScopeClass. En caso de no tener padre el campo estaría con un string vacío.



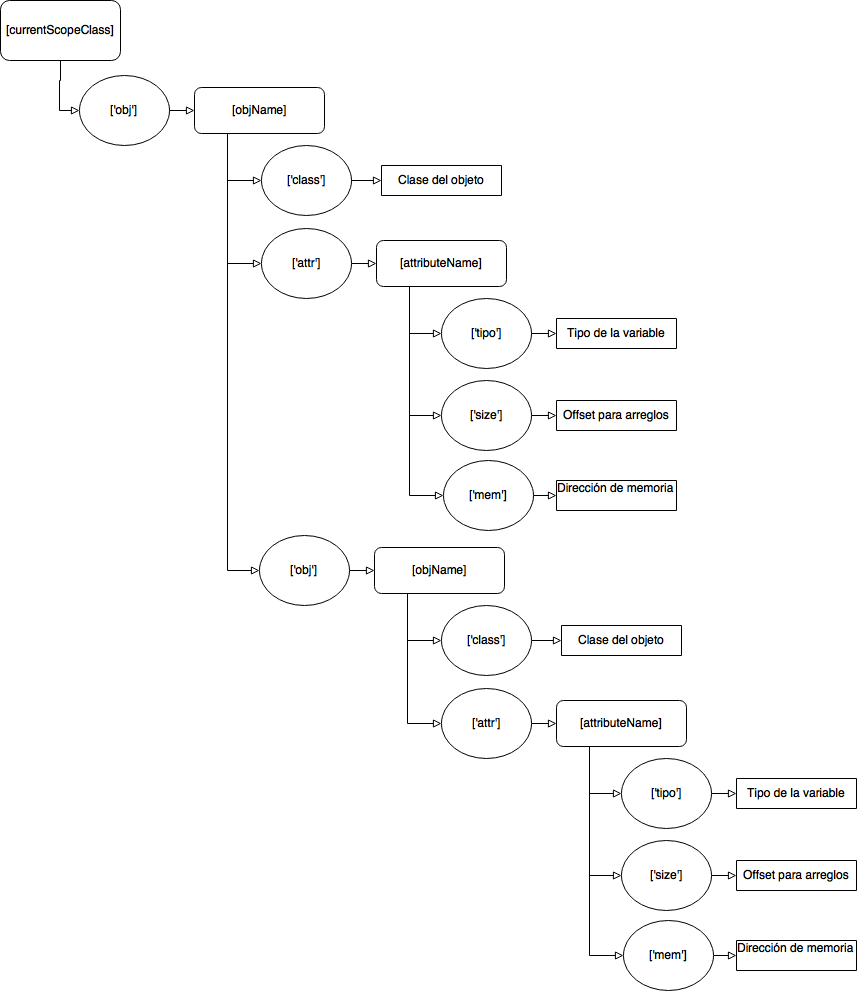
Con la llave ‘func’ se acceden a todas las funciones declaradas dentro del currentScopeClass. Estas funciones contiene la información que se muestra en el diagrama de la siguiente hoja. Varias y objetos guardados en este hash son de contexto local para la función dada (currentScopeFunction).



La 3ra llave de currentScopeClass es ‘vars’ la cual da acceso a las variables que viven dentro de todo el scope de clase actual. Las variables declaradas dentro de la función main serán guardadas en esta parte de la estructura, ya que no se maneja la función main() como un scope de función.

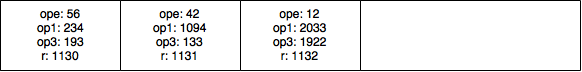


La última llave de currentScopeClass se encarga de manejar los objetos que existen dentro de la clase actual. [currentScopeClass][‘obj’][objName] cuenta con una llave extra de objeto para aceptar el caso de composición de objetos. Esta segunda llave obj solo podría existir si el currentScopeClass es ‘RedRobin’ (la clase principal) para evitar que se acepten composiciones de más niveles.



* 1. Cuadruplos

Para el manejo de cuadruplos se creó una clase Cuádruplo que tiene 4 atributos: op1, op2, ope y r, que corresponden al operando 1, operando 2, operador y el resultado. En el compilador se tiene una lista de cuadruplos donde se van agregando los cuádruplos creados.



* 1. Memorias virtuales

El manejo de memoria es segmentado. Se tiene cierto número de direcciones de memoria para cada tipo de dato y a la vez para cada scope de ese tipo de dato. Es decir, se tiene una dirección inicio y una dirección para números de clase, otras para números de función, etc.

Como se maneja en el compilador es un simple arreglo de enteros de 16 posiciones, una por cada tipo de dato con diferente scope.

memConts[]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 9 | 4 | 6 | 12 | 56 | 87 | 12 | 34 | 76 | 99 | 11 | 46 | 43 | 54 | 75 |

Para facilitar el uso de índices en el arreglo se ocupan diccionarios para obtener el índice que le corresponde a cierto tipo de variable con cierto scope. También se tienen diccionarios que guardan la dirección límite y la dirección de inicio de todos los tipos de memoria.

# Descripción de la máquina virtual

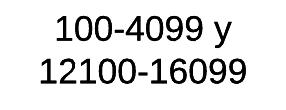
## Descripción del proceso de administración de memoria en ejecución

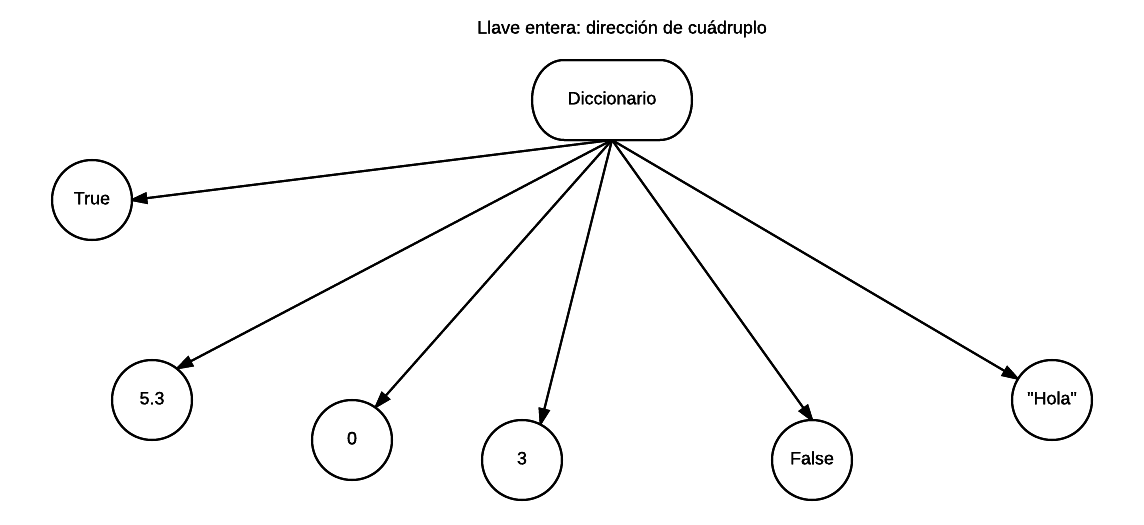
### Estructura de memoria global

Para el manejo de valores constantes y variables globales se utiliza un diccionario como la estructura de almacenamiento. Las llaves son las mismas direcciones que se obtienen de los cuádruplos y obtienen el valor almacenado. La gran ventaja de utilizar diccionarios es su fácil manejo de almacenamiento de valores de distintos tipos y que se pueden acceder a ellos en un tiempo promedio constante.

Esta estructura por lo mismo de ser global, se mantiene viva para toda la ejecución del programa en ejecución.

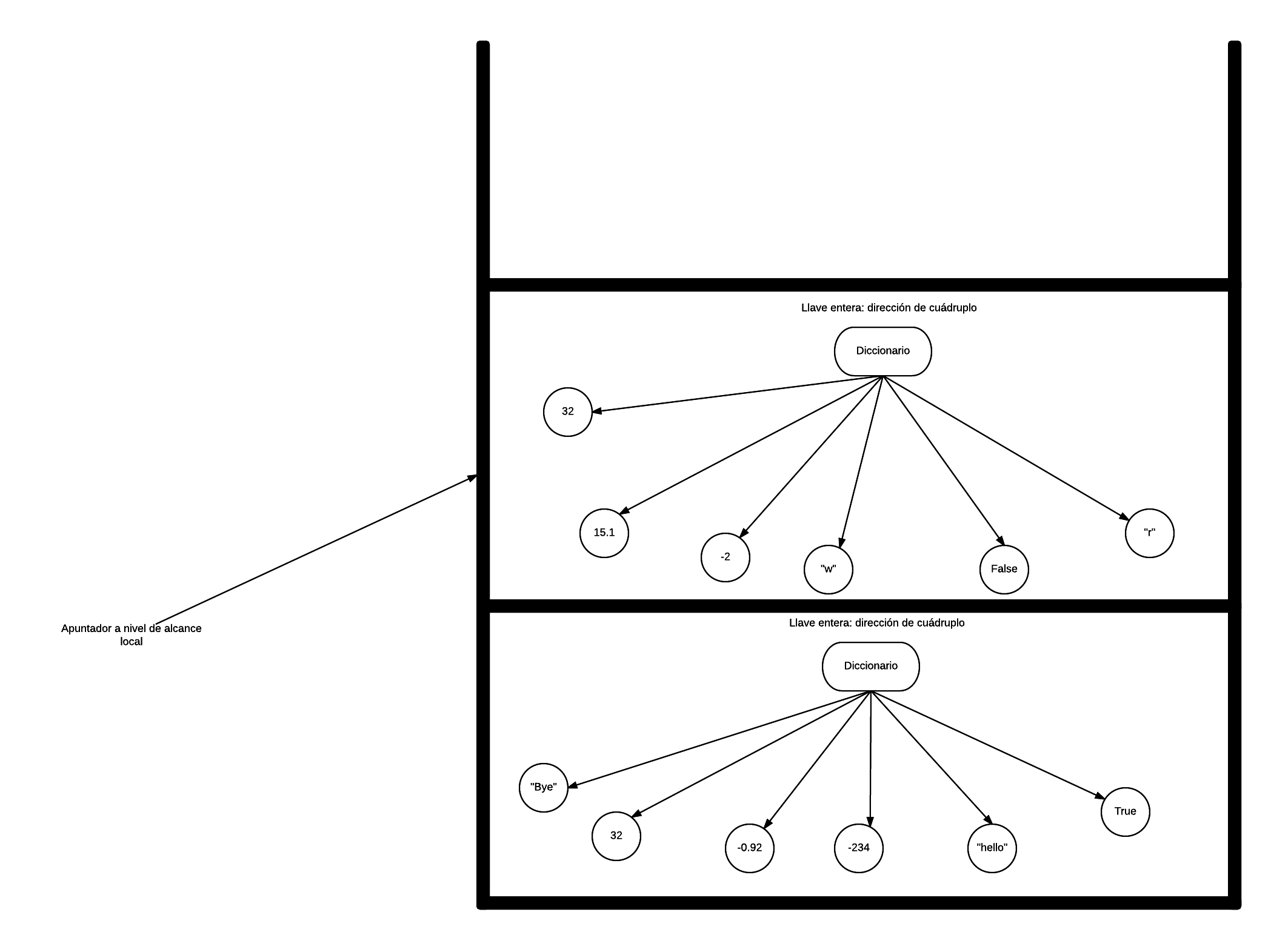
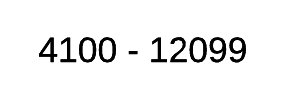
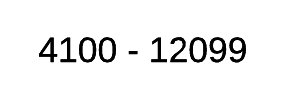
Direcciones





### Estructura de memoria local

La memoria local se maneja con una estructura de pilas que maneja los distintos niveles que pueden existir al realizar invocaciones de funciones que cuentan con un contexto local distinto. En cada elemento de pila se maneja un diccionario de la misma manera que la memoria global y se utiliza el mismo rango de instrucciones en cada una pero cada nivel almacena un distinto diccionario que representa un contexto de función invocada distinta. Se maneja una variable global que apunte al nivel de pila de contexto local sobre la que se está ejecutando.



### Asociación entre las direcciones virtuales de compilación y las reales de ejecución

Las direcciones obtenidas al leer el código objeto con la lista de cuádruplos son utilizados como valores llave para ambos tipos de estructura de datos una guardando las direcciones globales y otra manejando las direcciones locales. Para el caso de la estructura de pila de diccionarios de memoria local se tiene el apuntador al nivel de alcance local en la que se encuentra el programa. Por ello, al andar realizando operaciones incluyendo obtener valores de memoria y asignar nuevos valores, se identifica con el número de dirección a cuál estructura pertenece donde si la memoria está entre el rango 4100 y 12099, se está manejando un valor local.

Otra de las cosas a tomar en cuenta es que las direcciones indirectas eran indicados de forma negativa, es decir, cualquier dirección que fuera menor a 0 era manejado como una memoria indirecta.

# Pruebas del funcionamiento del lenguaje

## Factorial cíclico

* + 1. Código

class RedRobin {

RedRobin() {

number n, acum = 1;

read(n);

while(n >= 1) {

acum = acum \* n;

n = n - 1;

}

print(acum);

}

}

* + 1. Cuádruplos

2

1~12101.0

-1~12100.0

12

0~50~-1~-1~1

1~16~12101.0~-1~101.0

2~57~-1~-1~100.0

3~17~100.0~12101.0~11100.0

4~51~11100.0~-1~10

5~14~101.0~100.0~8100.0

6~16~8100.0~-1~101.0

7~13~100.0~12101.0~8101.0

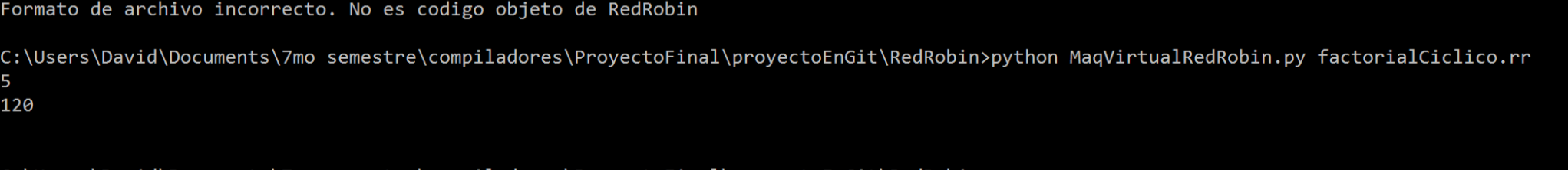
8~16~8101.0~-1~100.0

9~50~-1~-1~3

10~56~-1~-1~101.0

11~64~-1~-1~-1

* + 1. Resultados



## Factorial recursivo

* + 1. Código

class RedRobin {

func public number factorial(number: n) {

if (n == 0) {

give 1;

}

give n \* factorial(n - 1);

}

func public number factorialTerminal(number: n, acum) {

if (n == 0) {

give acum;

}

give factorialTerminal(n - 1, acum \* n);

}

RedRobin() {

number n;

read(n);

print(factorial(n));

print(factorialTerminal(n, 1));

}

}

* + 1. Cuádruplos

3

1~12102.0

-1~12100.0

0~12101.0

41

0~50~-1~-1~28

1~19~4100.0~12101.0~11100.0

2~51~11100.0~-1~5

3~62~12102.0~-1~100.0

4~63~-1~-1~-1

5~53~-1~-1~1

6~13~4100.0~12102.0~8100.0

7~55~8100.0~-1~4100.0

8~54~-1~-1~1

9~67~100.0~-1~8101.0

10~14~4100.0~8101.0~8102.0

11~62~8102.0~-1~100.0

12~63~-1~-1~-1

13~63~-1~-1~-1

14~19~4101.0~12101.0~11101.0

15~51~11101.0~-1~18

16~62~4102.0~-1~101.0

17~63~-1~-1~-1

18~53~-1~-1~14

19~13~4101.0~12102.0~8103.0

20~55~8103.0~-1~4101.0

21~14~4102.0~4101.0~8104.0

22~55~8104.0~-1~4102.0

23~54~-1~-1~14

24~67~101.0~-1~8105.0

25~62~8105.0~-1~101.0

26~63~-1~-1~-1

27~63~-1~-1~-1

28~57~-1~-1~102.0

29~53~-1~-1~1

30~55~102.0~-1~4100.0

31~54~-1~-1~1

32~67~100.0~-1~8106.0

33~56~-1~-1~8106.0

34~53~-1~-1~14

35~55~102.0~-1~4101.0

36~55~12102.0~-1~4102.0

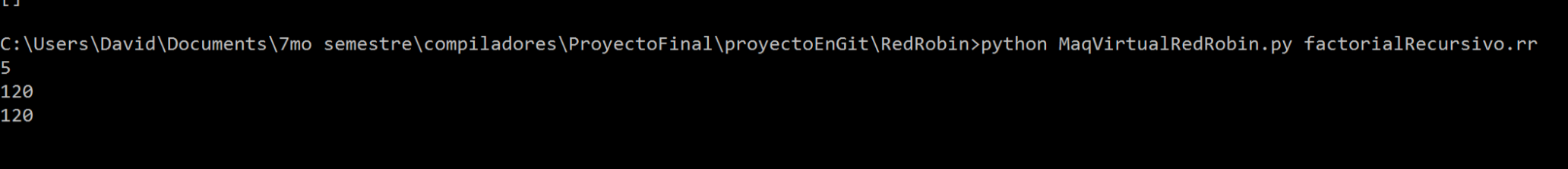
37~54~-1~-1~14

38~67~101.0~-1~8107.0

39~56~-1~-1~8107.0

40~64~-1~-1~-1

* + 1. Resultados



## Fibonacci cíclico

* + 1. Código

class RedRobin {

RedRobin() {

number n, aux, i;

number p = 0, q = 1;

read(n);

for i in (0 \_ n - 1) step 1 {

aux = p + q;

p = q;

q = aux;

}

print(p);

}

}

* + 1. Cuádruplos

3

0~12101.0

-1~12100.0

1~12102.0

17

0~50~-1~-1~1

1~16~12101.0~-1~103.0

2~16~12102.0~-1~104.0

3~57~-1~-1~100.0

4~16~12101.0~-1~102.0

5~13~100.0~12102.0~8100.0

6~18~102.0~8100.0~11100.0

7~51~11100.0~-1~15

8~12~103.0~104.0~8101.0

9~16~8101.0~-1~101.0

10~16~104.0~-1~103.0

11~16~101.0~-1~104.0

12~12~102.0~12102.0~8102.0

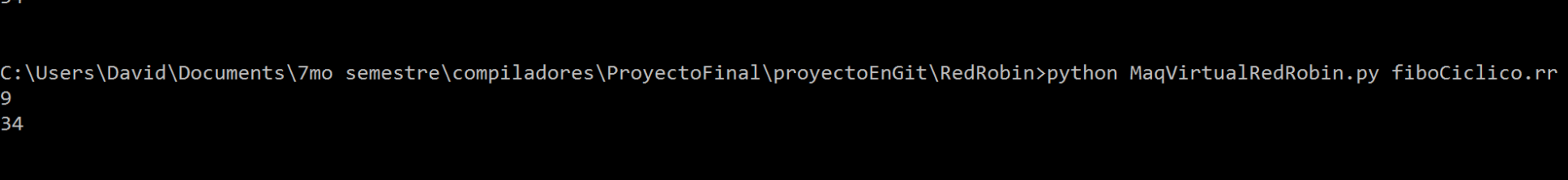
13~16~8102.0~-1~102.0

14~50~-1~-1~5

15~56~-1~-1~103.0

16~64~-1~-1~-1

* + 1. Resultados



## Fibonacci recursivo

* + 1. Código

class RedRobin {

func public number factorial(number: n) {

if (n == 0) {

give 1;

}

give n \* factorial(n - 1);

}

func public number factorialTerminal(number: n, acum) {

if (n == 0) {

give acum;

}

give factorialTerminal(n - 1, acum \* n);

}

RedRobin() {

number n;

read(n);

print(factorial(n));

print(factorialTerminal(n, 1));

}

}

* + 1. Cuádruplos

3

1~12102.0

-1~12100.0

0~12101.0

41

0~50~-1~-1~28

1~19~4100.0~12101.0~11100.0

2~51~11100.0~-1~5

3~62~12102.0~-1~100.0

4~63~-1~-1~-1

5~53~-1~-1~1

6~13~4100.0~12102.0~8100.0

7~55~8100.0~-1~4100.0

8~54~-1~-1~1

9~67~100.0~-1~8101.0

10~14~4100.0~8101.0~8102.0

11~62~8102.0~-1~100.0

12~63~-1~-1~-1

13~63~-1~-1~-1

14~19~4101.0~12101.0~11101.0

15~51~11101.0~-1~18

16~62~4102.0~-1~101.0

17~63~-1~-1~-1

18~53~-1~-1~14

19~13~4101.0~12102.0~8103.0

20~55~8103.0~-1~4101.0

21~14~4102.0~4101.0~8104.0

22~55~8104.0~-1~4102.0

23~54~-1~-1~14

24~67~101.0~-1~8105.0

25~62~8105.0~-1~101.0

26~63~-1~-1~-1

27~63~-1~-1~-1

28~57~-1~-1~102.0

29~53~-1~-1~1

30~55~102.0~-1~4100.0

31~54~-1~-1~1

32~67~100.0~-1~8106.0

33~56~-1~-1~8106.0

34~53~-1~-1~14

35~55~102.0~-1~4101.0

36~55~12102.0~-1~4102.0

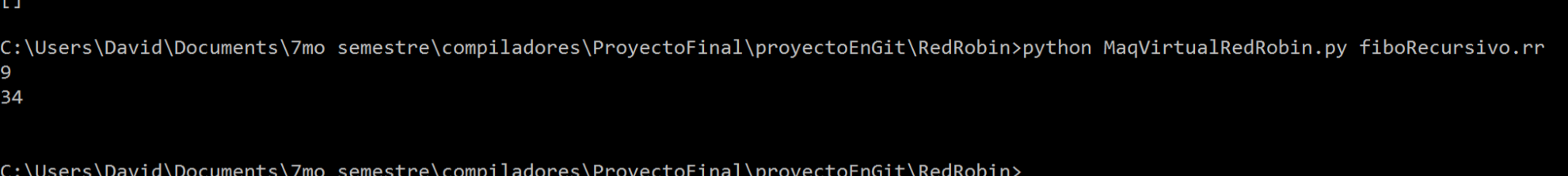
37~54~-1~-1~14

38~67~101.0~-1~8107.0

39~56~-1~-1~8107.0

40~64~-1~-1~-1

* + 1. Resultados



## Find en vector

* + 1. Código

class RedRobin {

RedRobin() {

number v[100], i, n;

// Llenamos con informacion el arreglo

for i in (0 \_ 99) step 1 {

v[i] = i \* 2;

}

read(n);

number pos = -1;

// busqueda del valor dado

for i in (0 \_ 99) step 1 {

if (v[i] == n) {

pos = i;

i = 100;

}

}

// imprimimos resultado

if (pos == -1) {

print("No encontrado");

}

else {

print("Valor encontrado en posicion " + toString(pos));

}

}

}

* + 1. Cuádruplos

9

100~12105.0

0~12102.0

"Valor encontrado en posicion "~14101.0

1~12104.0

"No encontrado"~14100.0

2~12106.0

99~12103.0

-1~12100.0

100~12101.0

35

0~50~-1~-1~1

1~16~12102.0~-1~200.0

2~18~200.0~12103.0~11100.0

3~51~11100.0~-1~11

4~66~200.0~0~99

5~12~12105.0~200.0~8100.0

6~14~200.0~12106.0~8101.0

7~16~8101.0~-1~-8100.0

8~12~200.0~12104.0~8102.0

9~16~8102.0~-1~200.0

10~50~-1~-1~2

11~57~-1~-1~201.0

12~14~12100.0~12104.0~8103.0

13~16~8103.0~-1~202.0

14~16~12102.0~-1~200.0

15~18~200.0~12103.0~11101.0

16~51~11101.0~-1~26

17~66~200.0~0~99

18~12~12105.0~200.0~8104.0

19~19~-8104.0~201.0~11102.0

20~51~11102.0~-1~23

21~16~200.0~-1~202.0

22~16~12101.0~-1~200.0

23~12~200.0~12104.0~8105.0

24~16~8105.0~-1~200.0

25~50~-1~-1~15

26~14~12100.0~12104.0~8106.0

27~19~202.0~8106.0~11103.0

28~51~11103.0~-1~31

29~56~-1~-1~14100.0

30~50~-1~-1~34

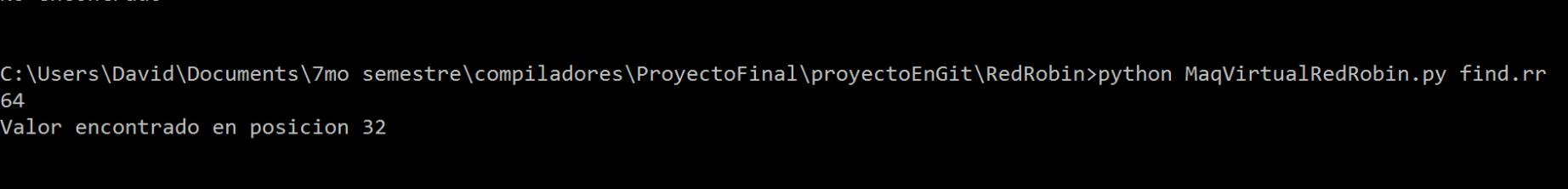
31~60~202.0~-1~10100.0

32~12~14101.0~10100.0~10101.0

33~56~-1~-1~10101.0

34~64~-1~-1~-1

* + 1. Resultados



## Sort de vector

* + 1. Código

class RedRobin {

class MergeSort {

public number v[100];

public number aux[100];

func public empty merge(number: ini, m, fin) {

number u = ini, i = ini, j = m + 1;

while(i <= m and j <= fin) {

if (v[i] <= v[j]) {

aux[u] = v[i];

i = i + 1;

}

else {

aux[u] = v[j];

j = j + 1;

}

u = u + 1;

}

for i in (i \_ m) step 1 {

aux[u] = v[i];

u = u + 1;

}

for j in (j \_ fin) step 1 {

aux[u] = v[j];

u = u + 1;

}

for i in (ini \_ fin) step 1 {

v[i] = aux[i];

}

}

func public empty divide(number: ini, fin) {

if (ini+1 <= fin) {

divide(ini, (ini+fin)/2);

divide((ini+fin)/2 + 1, fin);

merge(ini,(ini+fin)/2, fin);

}

}

}

RedRobin() {

MergeSort sort;

sort.v[0] = 4;

sort.v[1] = 10;

sort.v[2] = -2;

sort.v[3] = 1;

sort.v[4] = 0;

sort.divide(0, 4);

number i;

for i in (0 \_ 4) step 1 {

print(sort.v[i]);

}

}

}

* + 1. Cuádruplos

11

400~12107.0

10~12109.0

2~12105.0

100~12103.0

0~12106.0

200~12104.0

1~12102.0

4~12108.0

3~12110.0

-1~12100.0

100~12101.0

129

0~50~-1~-1~95

1~16~4100.0~-1~4103.0

2~16~4100.0~-1~4104.0

3~12~4101.0~12102.0~8100.0

4~16~8100.0~-1~4105.0

5~18~4104.0~4101.0~11100.0

6~18~4105.0~4102.0~11101.0

7~22~11100.0~11101.0~11102.0

8~51~11102.0~-1~33

9~66~4104.0~0~99

10~12~12103.0~4104.0~8101.0

11~66~4105.0~0~99

12~12~12103.0~4105.0~8102.0

13~18~-8101.0~-8102.0~11103.0

14~51~11103.0~-1~23

15~66~4103.0~0~99

16~12~12104.0~4103.0~8103.0

17~66~4104.0~0~99

18~12~12103.0~4104.0~8104.0

19~16~-8104.0~-1~-8103.0

20~12~4104.0~12102.0~8105.0

21~16~8105.0~-1~4104.0

22~50~-1~-1~30

23~66~4103.0~0~99

24~12~12104.0~4103.0~8106.0

25~66~4105.0~0~99

26~12~12103.0~4105.0~8107.0

27~16~-8107.0~-1~-8106.0

28~12~4105.0~12102.0~8108.0

29~16~8108.0~-1~4105.0

30~12~4103.0~12102.0~8109.0

31~16~8109.0~-1~4103.0

32~50~-1~-1~5

33~16~4104.0~-1~4104.0

34~18~4104.0~4101.0~11104.0

35~51~11104.0~-1~46

36~66~4103.0~0~99

37~12~12104.0~4103.0~8110.0

38~66~4104.0~0~99

39~12~12103.0~4104.0~8111.0

40~16~-8111.0~-1~-8110.0

41~12~4103.0~12102.0~8112.0

42~16~8112.0~-1~4103.0

43~12~4104.0~12102.0~8113.0

44~16~8113.0~-1~4104.0

45~50~-1~-1~34

46~16~4105.0~-1~4105.0

47~18~4105.0~4102.0~11105.0

48~51~11105.0~-1~59

49~66~4103.0~0~99

50~12~12104.0~4103.0~8114.0

51~66~4105.0~0~99

52~12~12103.0~4105.0~8115.0

53~16~-8115.0~-1~-8114.0

54~12~4103.0~12102.0~8116.0

55~16~8116.0~-1~4103.0

56~12~4105.0~12102.0~8117.0

57~16~8117.0~-1~4105.0

58~50~-1~-1~47

59~16~4100.0~-1~4104.0

60~18~4104.0~4102.0~11106.0

61~51~11106.0~-1~70

62~66~4104.0~0~99

63~12~12103.0~4104.0~8118.0

64~66~4104.0~0~99

65~12~12104.0~4104.0~8119.0

66~16~-8119.0~-1~-8118.0

67~12~4104.0~12102.0~8120.0

68~16~8120.0~-1~4104.0

69~50~-1~-1~60

70~63~-1~-1~-1

71~12~4106.0~12102.0~8121.0

72~18~8121.0~4107.0~11107.0

73~51~11107.0~-1~94

74~53~-1~-1~71

75~55~4106.0~-1~4106.0

76~12~4106.0~4107.0~8122.0

77~15~8122.0~12105.0~8123.0

78~55~8123.0~-1~4107.0

79~54~-1~-1~71

80~53~-1~-1~71

81~12~4106.0~4107.0~8124.0

82~15~8124.0~12105.0~8125.0

83~12~8125.0~12102.0~8126.0

84~55~8126.0~-1~4106.0

85~55~4107.0~-1~4107.0

86~54~-1~-1~71

87~53~-1~-1~1

88~55~4106.0~-1~4100.0

89~12~4106.0~4107.0~8127.0

90~15~8127.0~12105.0~8128.0

91~55~8128.0~-1~4101.0

92~55~4107.0~-1~4102.0

93~54~-1~-1~1

94~63~-1~-1~-1

95~66~12106.0~0~99

96~12~12107.0~12106.0~8129.0

97~16~12108.0~-1~-8129.0

98~66~12102.0~0~99

99~12~12107.0~12102.0~8130.0

100~16~12109.0~-1~-8130.0

101~66~12105.0~0~99

102~12~12107.0~12105.0~8131.0

103~14~12100.0~12105.0~8132.0

104~16~8132.0~-1~-8131.0

105~66~12110.0~0~99

106~12~12107.0~12110.0~8133.0

107~16~12102.0~-1~-8133.0

108~66~12108.0~0~99

109~12~12107.0~12108.0~8134.0

110~16~12106.0~-1~-8134.0

111~53~-1~-1~71

112~55~12106.0~-1~4106.0

113~55~12108.0~-1~4107.0

114~69~100.0~100~400.0

115~69~200.0~100~300.0

116~54~-1~-1~71

117~65~100.0~100~400.0

118~65~200.0~100~300.0

119~16~12106.0~-1~500.0

120~18~500.0~12108.0~11108.0

121~51~11108.0~-1~128

122~66~500.0~0~99

123~12~12107.0~500.0~8135.0

124~56~-1~-1~-8135.0

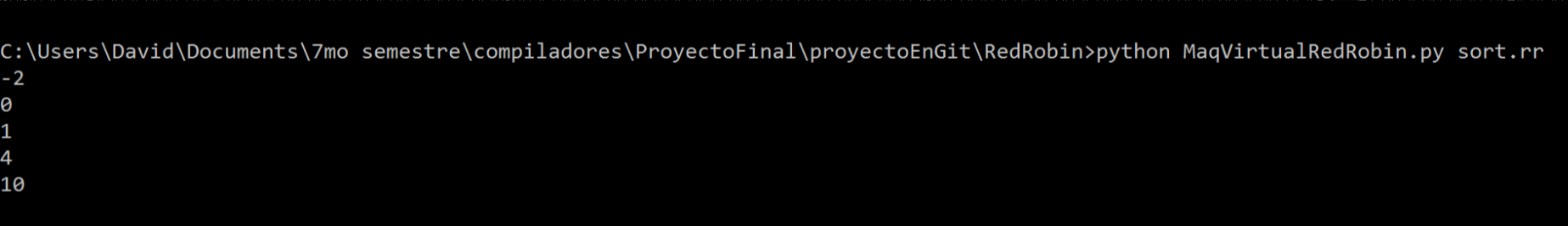
125~12~500.0~12102.0~8136.0

126~16~8136.0~-1~500.0

127~50~-1~-1~120

128~64~-1~-1~-1

* + 1. Resultados



## Pruebas propias

* + 1. Código

class RedRobin {

class Students {

public string name;

secret number edad;

func public empty showInfo() {

print("He's name is " + name + "and he's " + toString(edad));

}

func public number getEdad() {

give edad;

}

func public empty setEdad(number: newEdad) {

edad = newEdad;

}

}

class Interns inherit Students {

public number numberOfInternships;

secret string companias[5];

func public empty listarCompanias() {

number id;

print("Companias en las que ha realizado internships: " + name);

for id in (0 \_ 4) step 1 {

print(companias[id]);

}

}

func public empty poblarCompanias() {

companias[1] = "Microsoft";

companias[2] = "Facebook";

companias[4] = "Microsoft";

}

}

class Empresa {

public number ID;

secret string name;

secret Interns practicante;

func public empty datosIntern() {

print("Empresa: " + name);

print("Nombre de practicante: " + practicante.name);

print("Edad de practicante: " + toString(practicante.getEdad()));

print("Asignando nueva edad a practicante valida que tenga 23");

practicante.setEdad(23);

print("Nuevamente edad de practicante: " + toString(practicante.getEdad()));

practicante.showInfo();

practicante.listarCompanias();

}

func public bool inicializaIntern() {

practicante.name = "Omar Manjarrez";

practicante.numberOfInternships = 3;

practicante.poblarCompanias();

give true;

}

func public empty setName(string: nombre) {

name = nombre;

}

}

func public empty auxiliar() {

Students novato;

novato.name = "David";

novato.setEdad(21);

novato.showInfo();

}

RedRobin() {

Empresa internacional;

if (internacional.inicializaIntern()) {

internacional.datosIntern();

auxiliar();

}

}

}

* + 1. Cuádruplos

22

23~12107.0

2~12106.0

4~12103.0

"Empresa: "~14105.0

"Companias en las que ha realizado internships: "~14102.0

"He's name is "~14100.0

"Microsoft"~14103.0

"Asignando nueva edad a practicante valida que tenga 23"~14108.0

"David"~14111.0

"Nuevamente edad de practicante: "~14109.0

5~12101.0

"Facebook"~14104.0

21~12109.0

"Edad de practicante: "~14107.0

1~12104.0

"Nombre de practicante: "~14106.0

"and he's "~14101.0

"Omar Manjarrez"~14110.0

0~12102.0

-1~12100.0

2102~12105.0

3~12108.0

148

0~50~-1~-1~115

1~12~14100.0~2100.0~10100.0

2~12~10100.0~14101.0~10101.0

3~60~100.0~-1~10102.0

4~12~10101.0~10102.0~10103.0

5~56~-1~-1~10103.0

6~63~-1~-1~-1

7~62~100.0~-1~101.0

8~63~-1~-1~-1

9~63~-1~-1~-1

10~16~4100.0~-1~100.0

11~63~-1~-1~-1

12~12~14102.0~2101.0~10104.0

13~56~-1~-1~10104.0

14~16~12102.0~-1~4100.0

15~18~4100.0~12103.0~11100.0

16~51~11100.0~-1~23

17~66~4100.0~0~4

18~12~12105.0~4100.0~10105.0

19~56~-1~-1~-10105.0

20~12~4100.0~12104.0~8100.0

21~16~8100.0~-1~4100.0

22~50~-1~-1~15

23~63~-1~-1~-1

24~66~12104.0~0~4

25~12~12105.0~12104.0~10106.0

26~16~14103.0~-1~-10106.0

27~66~12106.0~0~4

28~12~12105.0~12106.0~10107.0

29~16~14104.0~-1~-10107.0

30~66~12103.0~0~4

31~12~12105.0~12103.0~10108.0

32~16~14103.0~-1~-10108.0

33~63~-1~-1~-1

34~12~14105.0~2107.0~10109.0

35~56~-1~-1~10109.0

36~12~14106.0~2113.0~10110.0

37~56~-1~-1~10110.0

38~53~-1~-1~7

39~69~2100.0~0~2113.0

40~69~100.0~0~106.0

41~54~-1~-1~7

42~65~2100.0~0~2113.0

43~65~100.0~0~106.0

44~67~101.0~-1~8101.0

45~60~8101.0~-1~10111.0

46~12~14107.0~10111.0~10112.0

47~56~-1~-1~10112.0

48~56~-1~-1~14108.0

49~53~-1~-1~10

50~55~12107.0~-1~4100.0

51~69~2100.0~0~2113.0

52~69~100.0~0~106.0

53~54~-1~-1~10

54~65~2100.0~0~2113.0

55~65~100.0~0~106.0

56~53~-1~-1~7

57~69~2100.0~0~2113.0

58~69~100.0~0~106.0

59~54~-1~-1~7

60~65~2100.0~0~2113.0

61~65~100.0~0~106.0

62~67~101.0~-1~8102.0

63~60~8102.0~-1~10113.0

64~12~14109.0~10113.0~10114.0

65~56~-1~-1~10114.0

66~53~-1~-1~1

67~69~2100.0~0~2113.0

68~69~100.0~0~106.0

69~54~-1~-1~1

70~65~2100.0~0~2113.0

71~65~100.0~0~106.0

72~53~-1~-1~12

73~69~103.0~0~105.0

74~69~102.0~0~106.0

75~69~2102.0~5~2108.0

76~69~2101.0~0~2113.0

77~54~-1~-1~12

78~65~103.0~0~105.0

79~65~102.0~0~106.0

80~65~2102.0~5~2108.0

81~65~2101.0~0~2113.0

82~63~-1~-1~-1

83~16~14110.0~-1~2113.0

84~16~12108.0~-1~105.0

85~53~-1~-1~24

86~69~103.0~0~105.0

87~69~102.0~0~106.0

88~69~2102.0~5~2108.0

89~69~2101.0~0~2113.0

90~54~-1~-1~24

91~65~103.0~0~105.0

92~65~102.0~0~106.0

93~65~2102.0~5~2108.0

94~65~2101.0~0~2113.0

95~62~15101.0~-1~3100.0

96~63~-1~-1~-1

97~63~-1~-1~-1

98~16~6100.0~-1~2107.0

99~63~-1~-1~-1

100~16~14111.0~-1~2114.0

101~53~-1~-1~10

102~55~12109.0~-1~4100.0

103~69~2100.0~0~2114.0

104~69~100.0~0~107.0

105~54~-1~-1~10

106~65~2100.0~0~2114.0

107~65~100.0~0~107.0

108~53~-1~-1~1

109~69~2100.0~0~2114.0

110~69~100.0~0~107.0

111~54~-1~-1~1

112~65~2100.0~0~2114.0

113~65~100.0~0~107.0

114~63~-1~-1~-1

115~53~-1~-1~83

116~69~2107.0~0~2115.0

117~69~2108.0~5~2116.0

118~69~2113.0~0~2121.0

119~69~104.0~0~108.0

120~69~105.0~0~109.0

121~69~106.0~0~110.0

122~54~-1~-1~83

123~65~2107.0~0~2115.0

124~65~2108.0~5~2116.0

125~65~2113.0~0~2121.0

126~65~104.0~0~108.0

127~65~105.0~0~109.0

128~65~106.0~0~110.0

129~67~3100.0~-1~11101.0

130~51~11101.0~-1~147

131~53~-1~-1~34

132~69~2107.0~0~2115.0

133~69~2108.0~5~2116.0

134~69~2113.0~0~2121.0

135~69~104.0~0~108.0

136~69~105.0~0~109.0

137~69~106.0~0~110.0

138~54~-1~-1~34

139~65~2107.0~0~2115.0

140~65~2108.0~5~2116.0

141~65~2113.0~0~2121.0

142~65~104.0~0~108.0

143~65~105.0~0~109.0

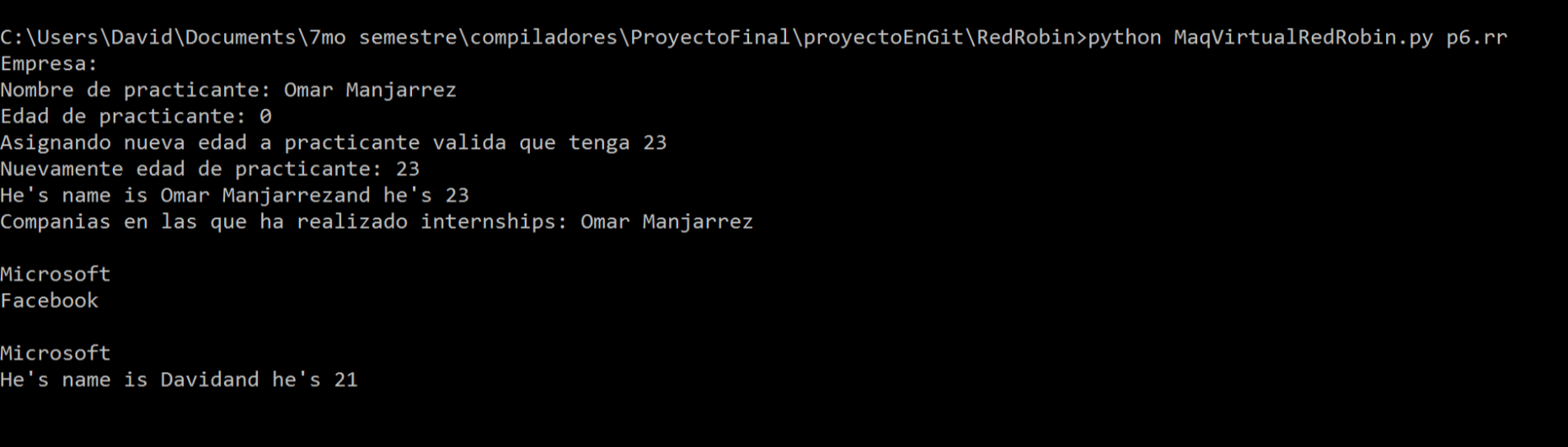
144~65~106.0~0~110.0

145~53~-1~-1~100

146~54~-1~-1~100

147~64~-1~-1~-1

* + 1. Resultados



# Listados documentados del proyecto

## Módulos importantes de semántica

En este módulo se hace la preparación para la memoria soporte un nuevo entorno para una clase que se está definiendo. Se valida que sea única, se identifican sus características como herencia o no de una clase padre y con ello obtener sus atributos.

# Llamada desde p\_clases

def p\_smnewclass(p):

'smnewclass :'

# Nuevo scope de clase, estoy definiendo una nueva calse

global currentScopeClass

newScopeClass = p[-2]

# Si ya existe, es error

if newScopeClass in dirProced:

terminate("REPEATED CLASS NAME")

else:

parent = ''

parentObjects = {}

parentVariables = {}

# Pregunto si tengo un padre

if p[-1] != None:

parent = p[-1]

# Si tengo un padre debo copiar todas su variables y sus objetos. Estos objetos no pueden tener mas objetos

# si mi padre ya tiene padre sería invalido

if dirProced[parent]['parent'] != "":

terminate("MORE THAN 1 LEVEL IN INHERITANCE")

# Jalo las variables de mi padre

parentVariables = copy.deepcopy(dirProced[parent]['vars'])

# Obtengo los objetos de mi padre, tampoco pueden tener mas objetos para evitar composicion de mas de 1 nivel

parentObjects = copy.deepcopy(dirProced[parent]['obj'])

# Valido lo anterior

# mi padre SI puede tener objetos, pero esos objetos NO pueden tener mas objetos

# por cada objeto reviso si en su definicion de clase tiene objetos

for objName in parentObjects:

objectClass = parentObjects[objName]['class']

if len(dirProced[objectClass]['obj']) > 0:

terminate("MORE THAN 1 LEVEL IN COMPOSITION2")

# Cada atributo de objeto debe tener direccion de memoria unica para mantener la unicidad de memorias dentro del scope

for objName in parentObjects:

dicAttr = parentObjects[objName]['attr']

for nameVar in dicAttr:

parentObjects[objName]['attr'][nameVar]['mem'] = getMemSpace(dicAttr[nameVar]['tipo'], 'Class', nameVar)

typeArray = parentObjects[objName]['attr'][nameVar]['tipo']

arraySize = parentObjects[objName]['attr'][nameVar]['size']

if int(arraySize) > 0:

memConts[memCont[typeArray + 'Class']] += (int(arraySize) - 1)

# Hago lo mismo de arriba con las variables

for varName in parentVariables:

parentVariables[varName]['mem'] = getMemSpace(parentVariables[varName]['tipo'], 'Class', varName)

typeArray = parentVariables[varName]['tipo']

arraySize = parentVariables[varName]['size']

if int(arraySize) > 0:

memConts[memCont[typeArray + 'Class']] += (int(arraySize) - 1)

dirProced[newScopeClass] = {'func': {}, 'vars': parentVariables, 'obj': parentObjects, 'parent': parent}

setScopeClass(newScopeClass)

Esta función se encarga de generar el espacio de memoria para una nueva variable en el contexto adecuado y valida que sea único su nombramiento respecto a nombres de funciones, clases y otras variables ya sea en su alcance local o globales.

# Llamada desde p\_declaracion y p\_masdeclaraciones

def p\_smnewvariable(p):

'smnewvariable : '

global lastPrivilage

newVarName = p[-1]

# Si el nobre ya existe, está repetido

if existsVar(newVarName) or existsObj(newVarName) or existsFunc(newVarName):

terminate("REPETAED VARIABLE NAME: " + newVarName)

# Si estamos dentro de una funcion, variable local

if currentScopeFunction != '':

if isAtomic(currentType):

# Instancio el primitivo de funcion

dirProced[currentScopeClass]['func'][currentScopeFunction]['vars'][newVarName] = {'tipo': currentType, 'size': 0, 'mem': getMemSpace(currentType, 'Func', newVarName), 'privilage': lastPrivilage}

else:

# es un objeto

# Valido que la clase del objeto exista

if currentType not in dirProced or currentType == 'RedRobin':

terminate("WRONG OBJECT TYPE")

# Depende si estoy en red robin o en otra clase el comportamiento cambia

if currentScopeClass == 'RedRobin':

# Es un objeto instanciado en funcion dentro de red robin

# Creamos instancia de la clase

dics = createInstancia(currentType)

dirProced[currentScopeClass]['func'][currentScopeFunction]['obj'][newVarName] = {'class': currentType, 'attr': dics['attrs'], 'obj': dics['objs'], 'privilage': lastPrivilage}

else:

# Si tiene mas objetos mi composicion, seria error

if len(dirProced[currentType]['obj']) > 0:

terminate("MORE THAN 1 LEVEL IN COMPOSITION1")

# Arrastro las variables de la clase que estoy "instanciando" en esta funcion

variables = copy.deepcopy(dirProced[currentType]['vars'])

# Asigno direcciones nuevas para mantener unicidad

for varName in variables:

variables[varName]['mem'] = getMemSpace(variables[varName]['tipo'], 'Class', varName)

arraySize = variables[varName]['size']

if int(arraySize) > 0:

memConts[memCont[variables[varName]['tipo'] + 'Class']] += (int(arraySize) - 1)

# Actualizo el directorio

dirProced[currentScopeClass]['func'][currentScopeFunctino]['obj'][newVarName] = {'class': currentType, 'attr': variables, 'privilage': lastPrivilage}

else:

# else- Si estamos fuera de una funcion

if isAtomic(currentType):

# instancio el primitivo de clase

dirProced[currentScopeClass]['vars'][newVarName] = {'tipo': currentType, 'size': 0, 'mem': getMemSpace(currentType, 'Class', newVarName), 'privilage': lastPrivilage}

else:

# Es un objeto

# Valido que la clase del objeto exista

if currentType not in dirProced or currentType == 'RedRobin':

terminate("WRONG OBJECT TYPE")

# Depende si estoy en la clase main red robin o en alguna otra clase el comportamiento cambia

if currentScopeClass == 'RedRobin':

# Es un objeto instanciado en main

# Hay que crear una instancia de la clase

dics = createInstancia(currentType)

dirProced['RedRobin']['obj'][newVarName] = {'class': currentType, 'attr': dics['attrs'], 'obj': dics['objs'], 'privilage': lastPrivilage}

else:

# Si tiene mas objetos mi composicion, seria error

if len(dirProced[currentType]['obj']) > 0:

terminate("MORE THAN 1 LEVEL IN COMPOSITION1")

# Arrastro las variables de la clase que estoy "instanciando" en esta clase

variables = copy.deepcopy(dirProced[currentType]['vars'])

# Asigno direcciones nuevas para mantener unicidad

for varName in variables:

variables[varName]['mem'] = getMemSpace(variables[varName]['tipo'], 'Class', varName)

arraySize = variables[varName]['size']

if int(arraySize) > 0:

memConts[memCont[variables[varName]['tipo'] + 'Class']] += (int(arraySize) - 1)

# Actualizo el directorio de variables de la clase "papa"

dirProced[currentScopeClass]['obj'][newVarName] = {'class': currentType, 'attr': variables, 'privilage': lastPrivilage}

setLastPrivilage('')

Maneja la creación de alcances para la nueva memoria local de una función invocada y determina en qué ambiente fue llamada la misma si se trata de una función llamada como método de un objeto o como parte de los métodos del alcance de la ejecución.

def p\_smNewInvocacion(p):

'smNewInvocacion :'

global contParam

global currentFunction

global currentClass

funName = p[-2]

objPath = p[-1]

if objPath != None:

# delego invocaciones de objeto a otra funcino

newInvocacionFuncDeObj(funName, objPath)

else:

if funName in dirProced[currentScopeClass]['func']:

# Funcion simple en el scope actual

generalInvocationRutine(funName, currentScopeClass, None)

elif dirProced[currentScopeClass]['parent'] != "" and funName in dirProced[ dirProced[currentScopeClass]['parent'] ]['func']:

# funcion heredada

generalInvocationRutine(funName, dirProced[currentScopeClass]['parent'], None)

# Pido por referencia todos los atributos de la clase donde originalmente esta la funcion

for varName in dirProced[currentClass]['vars']:

dirReal = dirProced[currentScopeClass]['vars'][varName]['mem']

hashRef[dirReal] = dirProced[currentClass]['vars'][varName]['mem']

hashRefTam[dirReal] = dirProced[currentScopeClass]['vars'][varName]['size']

# tambien pido los objetos que pueda tener la calse donde se definio la funcion

for objName in dirProced[currentClass]['obj']:

# recorro todas las variables atomicas del objeto actual

objClass = dirProced[currentClass]['obj'][objName]['class']

for attrName in dirProced[objClass]['vars']:

dirReal = dirProced[currentScopeClass]['obj'][objName]['attr'][attrName]['mem']

hashRef[dirReal] = dirProced[objClass]['vars'][attrName]['mem']

hashRefTam[dirReal] = dirProced[currentScopeClass]['obj'][objName]['attr'][attrName]['size']

else:

terminate("Function " + funName + " not declared")

La función valide idSemantics se encarga de validar que un identificador que se está intentando utilizar sea válido. También está función verifica en que scope se encunetra la variable (currentObjPath) y si se trata de un identificador de arreglo (currentArray). Teniendo estos datos la función hace los cuadruplos pertinentes y agrega la dirección de memoria a la pila de operandos.

# Llamada de p\_identificador

def validateIdSemantics(currentIdName, currentObjPath, currentArray):

# Voy a delegar orientado a objetos a otra funcion

if currentObjPath != None:

validateObjSemantics(currentIdName, currentObjPath, currentArray)

else:

if not existsVar(currentIdName):

terminate("Variable " + currentIdName + " not declared")

# validar que no sea una varible de retorno de funcino

if currentIdName in dirProced[currentScopeClass]['func']:

terminate("Variable " + currentIdName + " is a function")

# si es arreglo hacer validaciones

if currentArray == '[':

# siempre sera accesible

arrayRutine(currentIdName, None, None)

else:

# variable valida, insertar a pila

success = False

if currentIdName in dirProced[currentScopeClass]['vars']:

# Estan siendo utilizadas dentro de su contexto, no ocupo privilegios

stackDirMem.append(dirProced[currentScopeClass]['vars'][currentIdName]['mem'])

elif currentScopeFunction == '' or currentIdName in dirProced[currentScopeClass]['func'][currentScopeFunction]['vars']:

# las variables locales no tienen privilegios

stackDirMem.append(dirProced[currentScopeClass]['func'][currentScopeFunction]['vars'][currentIdName]['mem'])

else:

terminate("variable " + currentIdName + " not found")

## Módulos importantes de máquina virtual

Este módulo es el más utilizado a lo largo de toda la ejecución de la máquina virtual pues recibe una dirección y nivel de alcance local actual. Lo que se busca retornar es el valor según la dirección. Entre los casos que checa es que si se trata de una dirección indirecta se invoca a sí misma enviando el absoluto de la dirección que es como estaba representado el indirecto. Al obtener el valor de la verdadera dirección, verifica si esta existe en el diccionario para retornar ese valor. De no existir el valor se inicializa en valor default según el tipo de valor siempre y cuando el código de operación del cuádruplo no sea de un retorno que indica que no hubo valor regresado por función.

#Recibe una direccion y nivel de alcance para retornar el valor de la estructura pertinente segun la misma direccion: memoria global o local

def findValueInMemory(numDireccion, alcanceFuncion):

#Se verifica si es una direccion indirecta

if numDireccion < -1:

#Se obtiene la verdadera direccion invocandose a si mismo

numDireccion = findValueInMemory(abs(numDireccion), alcanceFuncion)

if numDireccion < memStart['numberFunc'] or numDireccion > memLimit['boolTemp']:

#Se trata de una direccion global

if numDireccion in memEjecucion:

#Valor esta incializado

return memEjecucion[numDireccion]

elif liCuadruplos[apunCuadruplo].ope == 67:

#Si no existe valor en diccionario para operacion de asignar valor de retorno, significa que no hubo valor de retorno

terminate("Execution error. No returned value from method")

else:

#De ser cualquier otra operacion. Se verifica el tipo de direccion y se inicializa a su valor por default

if getTypeCode(numDireccion) == toCode['number']:

memEjecucion[numDireccion] = 0

return 0

elif getTypeCode(numDireccion) == toCode['real']:

memEjecucion[numDireccion] = 0.0

return 0.0

elif getTypeCode(numDireccion) == toCode['bool']:

memEjecucion[numDireccion] = False

return False

else:

memEjecucion[numDireccion] = ""

return ""

else:

#Se trata de una direccion local o temporal

if numDireccion in pilaMemoriaLocal[alcanceFuncion]:

#Valor esta inicializado

return pilaMemoriaLocal[alcanceFuncion][numDireccion]

elif liCuadruplos[apunCuadruplo].ope == 67:

#Si no existe valor en diccionario para operacion de asignar valor de retorno, significa que no hubo valor de retorno

terminate("Execution error. No returned value from method")

else:

#De ser cualquier otra operacion. Se verifica el tipo de direccion y se inicializa a su valor por default

if getTypeCode(numDireccion) == toCode['number']:

pilaMemoriaLocal[alcanceFuncion][numDireccion] = 0

return 0

elif getTypeCode(numDireccion) == toCode['real']:

pilaMemoriaLocal[alcanceFuncion][numDireccion] = 0.0

return 0.0

elif getTypeCode(numDireccion) == toCode['bool']:

pilaMemoriaLocal[alcanceFuncion][numDireccion] = False

return False

else:

pilaMemoriaLocal[alcanceFuncion][numDireccion] = ""

return ""

El método era es importante pues genera el nuevo espacio de memoria local que se va a utilizar antes de toda la preparación de envío de parámetros. La rutina es simple pues sólo se agrega un diccionario a la pila de diccionarios.

def era():

global pilaMemoriaLocal

# Se le agrega una nuevo entorno de variables locales para la funcion a ser invocada

pilaMemoriaLocal.append({})

El cuádruplo de “ref” o referencia se utiliza para asignar valores del alcance actual con los valores con los que terminaron sus referenciados en el entorno de almacenamiento local en la función recién invocada. Por ésta transición de los datos es importante no destruir la memoria local ya terminada de la ejecución de la función hasta asegurarse que ya no hay cuádruplos de referencia posteriores. De ser así se elimina el diccionario que está en el tope de la pila.

def referencia():

global pilaMemoriaLocal

operando1 = liCuadruplos[apunCuadruplo].op1

operando2 = liCuadruplos[apunCuadruplo].op2

#Se obtiene la direccion absoluta en caso de que sea una direccion indirecta en el entorno local actual

direccionAlmacenar = findAbsoluteAddress(liCuadruplos[apunCuadruplo].r, nivelAlcance)

#Se itera sobre el operando2 que indica si se trata de un arreglo en caso de ser mayor a 0

if operando2 == 0:

#El valor a pasarse como referencia NO es una estructura pero igual debe ejecutar una vez el ciclo

operando2 = 1

#Se tranfieren los valores de referencia sobre todos los elementos en caso de ser un arreglo o solo una vez

for ite in range(0, operando2):

#Se busca valor en el entorno de memoria local que se acaba de dejar de la funcion que se termino de invocar

valor1 = findValueInMemory(operando1 + ite, nivelAlcance + 1)

direccionActual = direccionAlmacenar + ite

if isGlobal(direccionActual):

#Se checa si la direccion a asignar valor es entera para almacenar solo la parte entera

if isNumber(direccionActual):

memEjecucion[direccionActual] = int(valor1)

else:

memEjecucion[direccionActual] = valor1

else:

#Se checa si la direccion a asignar valor es entera para almacenar solo la parte entera

if isNumber(direccionActual):

pilaMemoriaLocal[nivelAlcance][direccionActual] = int(valor1)

else:

pilaMemoriaLocal[nivelAlcance][direccionActual] = valor1

#Se verifica si el siguiente NO es un cuadruplo de referencia

if liCuadruplos[apunCuadruplo + 1].ope != 65:

#Se 'destruye' toda la memoria de entorno local de la funcion que recien acaba de terminar pues ya no se requiere

pilaMemoriaLocal.pop()

La entrada de datos de teclado involucra realizar varias validaciones al castear el valor ingresado. Se tiene que verificar la dirección a la que se va a almacenar el valor ingresado. Para ello se debe obtener el tipo de valor que guarda la dirección y de acuerdo a eso castear al valor requerido pero escuchando a posibles excepciones que puedan ser arrojados como cuando el usuario ingresa caracteres especiales o letras cuando se esperaba un número.

def lecturaTeclado():

#Se obtiene la direccion absoluta en caso de que sea una direccion indirecta

direccionAlmacenar = findAbsoluteAddress(liCuadruplos[apunCuadruplo].r, nivelAlcance)

respuestaUsuario = input()

#En caso de excepcion arrojada esta variable indica a que se trato de castear el valor de usuario

tipoID = 0

try:

#Se identifica el tipo de direccion para saber al tipo de valor que se debe castear el valor ingresado por el usuario

if getTypeCode(direccionAlmacenar) == toCode['number']:

tipoID = 0

valor = int(respuestaUsuario)

elif getTypeCode(direccionAlmacenar) == toCode['real']:

tipoID = 1

valor = float(respuestaUsuario)

elif getTypeCode(direccionAlmacenar) == toCode['bool']:

tipoID = 2

if respuestaUsuario == 'true':

valor = True

elif respuestaUsuario == 'false':

valor = False

else:

#No se puede castear a booleano

terminate("Execution error. Cannot convert value to bool.")

else:

tipoID = 3

#Se almacena un string directo

valor = respuestaUsuario

except:

#Se lanzo una excepcion al tratar de castear el valor

if tipoID == 0:

terminate("Execution error. Cannot convert value to number.")

elif tipoID == 1:

terminate("Execution error. Cannot convert value to real.")

elif tipoID == 2:

terminate("Execution error. Cannot convert value to bool.")

else:

terminate("Execution error. Cannot convert value to string.")

#Se almacena valor en estructura correcta

if isGlobal(direccionAlmacenar):

#Se almacena valor global dentro de la estructura que maneja almacenamiento global

memEjecucion[direccionAlmacenar] = valor

else:

#Se almacena valor local dentro de la estructura que maneja almacenamiento local

pilaMemoriaLocal[nivelAlcance][direccionAlmacenar] = valor

En esta funcion se realiza algo similar a referencia pero en sentido contrario. Las direcciones del alcance actual son las que copian sus valores a las direcciones locales para el nuevo alcance que se va a manejar

def asignaRetorno():

operando1 = liCuadruplos[apunCuadruplo].op1

valor1 = findValueInMemory(operando1, nivelAlcance)

#Se obtiene la direccion absoluta en caso de que sea una direccion indirecta

direccionAlmacenar = findAbsoluteAddress(liCuadruplos[apunCuadruplo].r, nivelAlcance)

if isGlobal(direccionAlmacenar):

#Se checa si la direccion a asignar valor es entera para almacenar solo la parte entera

if isNumber(direccionAlmacenar):

memEjecucion[direccionAlmacenar] = int(valor1)

else:

memEjecucion[direccionAlmacenar] = valor1

else:

#Se checa si la direccion a asignar valor es entera para almacenar solo la parte entera

if isNumber(direccionAlmacenar):

pilaMemoriaLocal[nivelAlcance][direccionAlmacenar] = int(valor1)

else:

pilaMemoriaLocal[nivelAlcance][direccionAlmacenar] = valor1

## Manual de usuario

El manual de usuario consta de dos partes:

* Video demo: muestra cómo funciona el proceso de compilación, los comandos que se deben usar y la estructura básica de un programa en red robin. El video se encuentra anexado al CD de la entrega o en la siguiente liga: <https://youtu.be/GHz0f8wllwU>
* Quick reference: la guía rápida para empezar a programar en Red robin se encuentra en el repositorio de github en el README. Se puede accessar desde el siguiente enlace: <https://github.com/Pando65/RedRobin>