**Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Monterrey**

Diseño de compiladores

Maestra: Elda Quiroga

José Manuel González Castro A01280106

Raúl Axel Suárez Martínez A01195895

Fecha de entrega: miércoles 4 de mayo de 2016

Lenguaje orientado a objetos (Objective Plox)

**Descripción y documentación del proyecto**

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

José Manuel González Castro Raúl Axel Suárez Martínez

Tabla de contenidos:

[A) Descripción del Proyecto 4](#_Toc450076864)

[A.1) Visión, Objetivos, y Alcance del Proyecto 4](#_Toc450076865)

[A.2) Análisis de Requerimientos y Casos de Uso generales 4](#_Toc450076866)

[A.3) Descripción de los principales Test Cases 9](#_Toc450076867)

[A.4) Descripción del proceso de desarrollo 10](#_Toc450076868)

[Reflexión José Manuel: 11](#_Toc450076869)

[Reflexión Axel Suárez: 11](#_Toc450076870)

[B) Descripción del Lenguaje 12](#_Toc450076871)

[B.1) Nombre del lenguaje 12](#_Toc450076872)

[B.2) Descripción de las principales características del lenguaje 12](#_Toc450076873)

[B.3) Descripción de los posibles errores 12](#_Toc450076874)

[C) Descripción del Compilador 13](#_Toc450076875)

[C.1) Equipo de cómputo, lenguaje y utilerías usadas en el proyecto 13](#_Toc450076876)

[C.2) Descripción del Análisis Léxico 13](#_Toc450076877)

[C.2.1) Patrones de construcción de los elementos principales 13](#_Toc450076878)

[C.2.2) Tokens del lenguaje y código asociado 14](#_Toc450076879)

[C.3) Descripción del Análisis de Sintaxis 15](#_Toc450076880)

[C.3.1) Gramática Formal 15](#_Toc450076881)

[C.4) Descripción de Generación de Código Intermedio y Análisis Semántico 17](#_Toc450076882)

[C.4.1) Código de operación y direcciones virtuales asociadas 17](#_Toc450076883)

[C.4.2) Diagramas de sintaxis con las acciones correspondientes 18](#_Toc450076884)

[C.4.3) Descripción de las acciones semánticas y de código 24](#_Toc450076885)

[C.4.4) Tabla de consideraciones semánticas 25](#_Toc450076886)

[C.5) Descripción detallada del proceso de Administración de Memoria 26](#_Toc450076887)

[C.5.1) Especificación gráfica de las estructuras de datos utilizadas 26](#_Toc450076888)

[D) Descripción de la Máquina Virtual 31](#_Toc450076889)

[D.1) Equipo de cómputo, lenguaje y utilerías especiales utilizadas 31](#_Toc450076890)

[D.2) Arquitectura del proceso de Administración de Memoria en ejecución 32](#_Toc450076891)

[D.2.1) Especificación gráfica de las estructuras de datos utilizadas 32](#_Toc450076892)

[D.2.2) Asociación hecha entre las direcciones virtuales y reales 32](#_Toc450076893)

[D.2.3) Código con ejemplos de acceso a memoria 32](#_Toc450076894)

[E) Pruebas del funcionamiento del Lenguaje 32](#_Toc450076895)

[E.1) Pruebas que comprueban el funcionamiento del proyecto 32](#_Toc450076896)

[F) Listados documentados del proyecto 47](#_Toc450076897)

[Directorio de procedimientos 47](#_Toc450076898)

[Arreglos 48](#_Toc450076899)

[Validaciones semánticas de herencia 50](#_Toc450076900)

[Funciones 51](#_Toc450076901)

[Constantes, direcciones de memoria y validación del cubo semántico 52](#_Toc450076902)

[Recuperación de valores 53](#_Toc450076903)

[Máquina virtual 54](#_Toc450076904)

# Descripción del Proyecto

## A.1) Visión, Objetivos, y Alcance del Proyecto

**Visión:** El propósito de este proyecto fue crear un compilador para un lenguaje básico orientado a objetos llamado Objective Plox. Esto implicó pasar por fases desde el análisis léxico hasta la generación de una máquina virtual. Asimismo, el proyecto cuenta con documentación que se divide en dos partes. La primera de estas siendo la parte técnica que explica el proyecto en sí; descripciones del lenguaje, compilador, máquina virtual, pruebas y código documentado. Por el otro lado, la segunda parte cuenta con un manual de usuario que sirva como apoyo para programadores que utilicen el lenguaje.

**Objetivo:** Este proyecto tuvo como objetivo el crear un lenguaje con la funcionalidad básica de un lenguaje orientado a objetos. Esto significa que cuenta con estatutos condicionales, ciclos, métodos/funciones, clases, herencia, arreglos, etc., todo esto con el propósito de tener un lenguaje que pueda servir como una herramienta de aprendizaje para aquellas personas interesadas en conocer un poco sobre este paradigma de programación. Debido a esto el lenguaje cuenta con una sintaxis muy similar a la de otros lenguajes ya conocidos como Java, con la diferencia de que está más limitado debido al tiempo y que también cuenta con un léxico diferente que es más divertido e interesante para quienes desarrollen en él.

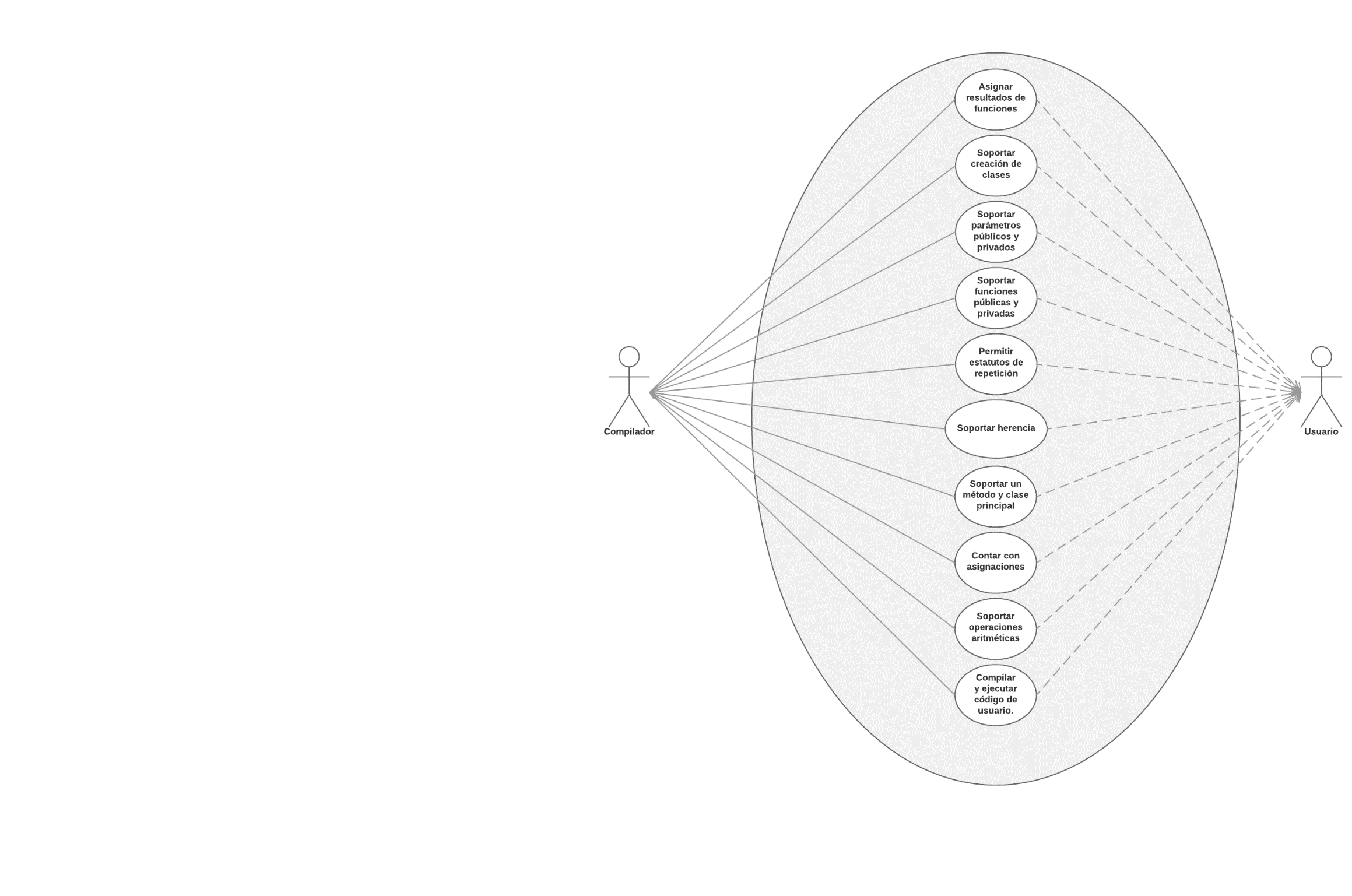
**Alcance del proyecto:** Crear un compilador para un lenguaje orientado a objetos que cuenta con la funcionalidad básica de estos, es decir, que cuenta con arreglos, funciones, clases, herencia, estatutos básicos como condicionales, ciclos y las operaciones que permitiría un lenguaje orientado a objetos como Java.

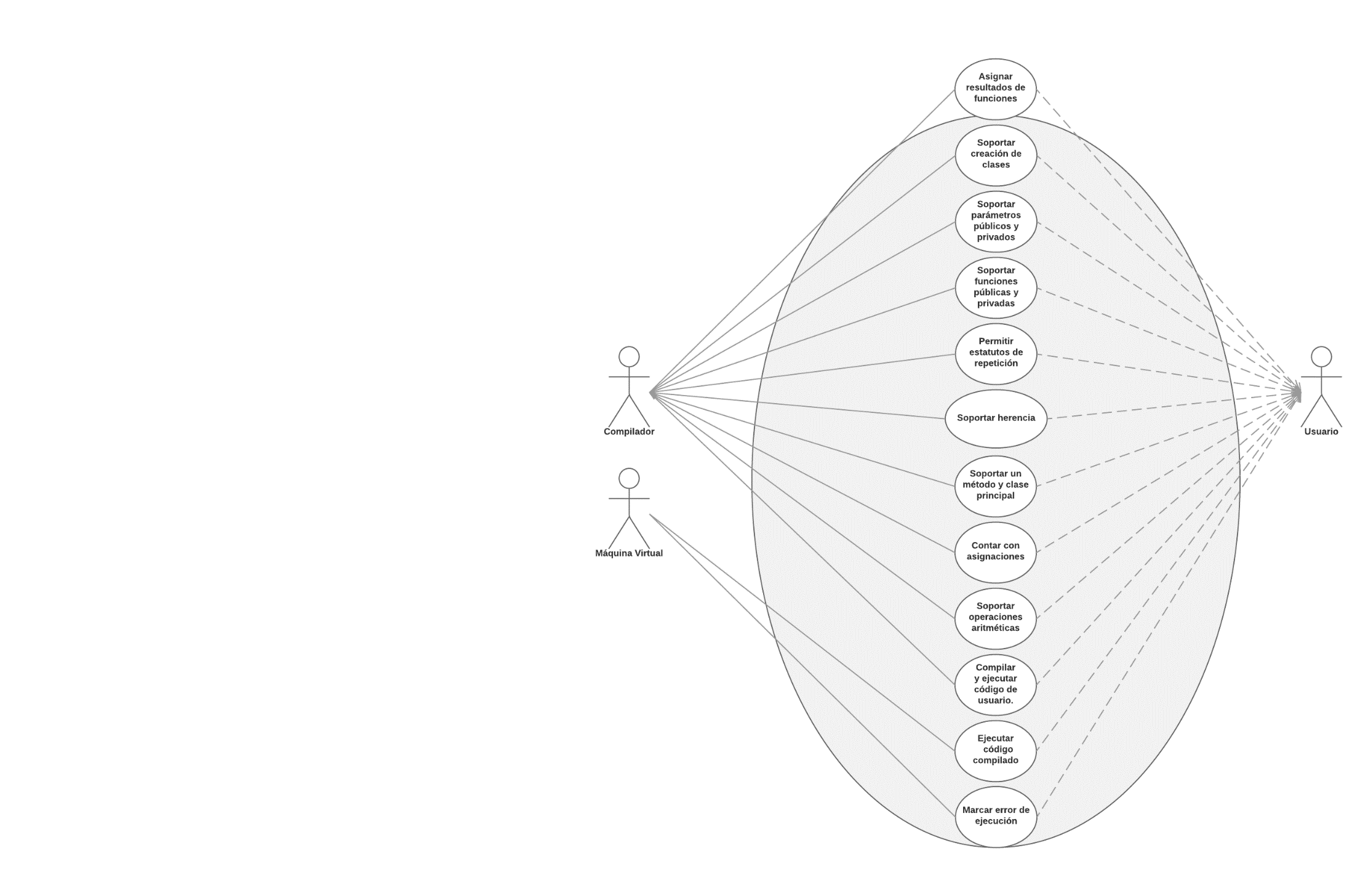
## A.2) Análisis de Requerimientos y Casos de Uso generales

El compilador a desarrollar debe de cumplir con ciertos requerimientos concernientes a los servicios que proporciona, indicando cómo debería reaccionar frente a entradas particulares y cómo debería de comportarse en determinadas situaciones. Los requerimientos funcionales elaborados para este proyecto son los siguientes:

**Requerimientos funcionales**:

* *RF01*: El compilador deberá marcar los errores durante la compilación.
* *RF02*: El compilador deberá terminar tan pronto encuentre un error y despliegue el mensaje correspondiente.
* *RF03*: El compilador deberá leer el código del programador de un archivo de texto.
* *RF04*: El compilador deberá permitir que se declaren variables de diferentes tipos de datos (number, decimal, char, string y logic).
* *RF05*: El compilador deberá permitir que se declaren arreglos con los diferentes tipos de datos (number, decimal, char, string y logic).
* *RF06*: El compilador deberá permitir el uso de estatutos condicionales (if, else, else if, unless).
* *RF07*: El compilador deberá permitir que el programador pueda imprimir información.
* *RF08*: El compilador deberá permitir que se puedan ingresar valores por medio de escritura en el teclado.
* *RF09*: El compilador deberá soportar la creación de funciones, tanto void como de los diferentes tipos de datos (number, decimal, char, string y logic).
* *RF10*: El compilador deberá permitir la llamada de funciones.
* *RF11*: El compilador deberá permitir la asignación del resultado de una función a una variable.
* *RF12*: El compilador deberá permitir la creación de clases.
* *RF13*: El compilador deberá permitir la creación de parámetros públicos y privados dentro de las clases.
* *RF14*: El compilador deberá permitir la creación de funciones públicas y privadas, con o sin retorno dentro de las clases.
* *RF15*: El compilador deberá permitir el uso de estatutos de repetición como ciclos while y do while.
* *RF16*: El compilador deberá permitir el uso de herencia entre clases.
* *RF17*: El compilador deberá contar con una clase principal y un método principal. (chief, equivalente al main).
* *RF18*: El compilador deberá permitir asignaciones básicas. (=, +=, \*=, -=, /=, ||=, &&=)
* *RF19*: El compilador deberá permitir realizar operaciones aritméticas básicas por medio de los signos +, -, /, \*.
* *RF20*: Compilar el código del usuario mostrar un mensaje indicando que no hubo errores.
* *RF21*: Ejecutar el código ya compilado con una máquina virtual sin marcar errores.
* *RF22*. Ejecutar el código ya compilado con una máquina virtual marcando errores y desplegando un mensaje apropiado para esto.

**Diagrama de casos de uso:**

****

**Principales casos de uso:**

|  |  |
| --- | --- |
| Nombre | Compilar un código con errores y mostrar el error. |
| Identificador | CU01 |
| Requerimiento Funcional | RF01, RF02 |
| Propósito | Terminar la ejecución de un programa y marcar el mensaje de error. |
| Precondiciones | El usuario ya cuenta con un archivo de texto que contiene el código del programa a compilar. |
| Postcondiciones | Se despliega un mensaje indicando el error y la línea de código donde tronó el programa. |
| Flujo normal | 1. Se ingresa el nombre del archivo de texto con el programa. 2. El programa empieza a compilar. 3. Se detecta un error durante la compilación y se aborta lo que resta de compilación. 4. Se despliega un mensaje indicando que hubo error durante la compilación y se muestro junto a la línea donde este fue encontrado. |
| Flujos alternativos | Ninguno |

|  |  |
| --- | --- |
| Nombre | Compilar un código sin errores durante compilación. |
| Identificador | CU02 |
| Requerimiento Funcional | RF03, RF04, RF05, RF06, RF07, RF08, RF09, RF10, RF11, RF12, RF13, RF14, RF15, RF16, RF17, RF18, RF19, RF20 |
| Propósito | Compilar y ejecutar un programa que cuente con todo lo que se solicita y que lo haga de manera exitosa. |
| Precondiciones | El usuario ya cuenta con un archivo de texto que contiene el código del programa a compilar. |
| Postcondiciones | Se despliega un mensaje indicando que la compilación fue exitosa. |
| Flujo normal | 1. Se ingresa el nombre del archivo de texto con el programa. 2. Se compila el programa, pasa por el análisis léxico, sintáctico y semántico. 3. Se crean los archivos necesarios a pasar a la máquina virtual. 4. Muestra un mensaje indicando que la compilación fue exitosa. |
| Flujos alternativos | Ninguno |

|  |  |
| --- | --- |
| Nombre | Ejecutar un código con errores |
| Identificador | CU03 |
| Requerimiento Funcional | RF22 |
| Propósito | Ejecutar un código que ya pasó por compilación pero que tiene errores de ejecución. |
| Precondiciones | El usuario ya compiló el archivo con el código del programa y no se encontraron errores. |
| Postcondiciones | Se despliega un mensaje indicando el primer error y termina la ejecución. |
| Flujo normal | 1. Se muestra un mensaje indicando que ya está en la fase de ejecución. 2. Se empieza a ejecutar el programa. 3. Se encuentra un error y se despliega un mensaje indicando el tipo de error encontrado. 4. Termina el proceso de ejecución del programa. |
| Flujos alternativos | Ninguno |

|  |  |
| --- | --- |
| Nombre | Ejecutar un código sin errores |
| Identificador | CU04 |
| Requerimiento Funcional | RF21 |
| Propósito | Ejecutar un código que ya pasó por compilación y que no tiene errores de ejecución. |
| Precondiciones | El usuario ya compiló el archivo con el código del programa y no se encontraron errores. |
| Postcondiciones | Se despliega lo que sea que imprima el programa, suponiendo que este tiene impresiones dentro del código. |
| Flujo normal | 1. Se muestra un mensaje indicando que ya está en la fase de ejecución. 2. Se empieza a ejecutar el programa. 3. Hace todos los accesos a memoria en base a los cuádruplos sin error alguno. 4. Hace las acciones correspondientes a los cuádruplos. 5. Termina la ejecución. |
| Flujos alternativos | Ninguno |

## A.3) Descripción de los principales Test Cases

Para poder probar el compilador y verificar que cuente con toda la funcionalidad establecida originalmente, se diseñarán casos de prueba que puedan tratar de atacar todo esto. Los casos de prueba consisten en programas diseñados para hacer que el compilador truene y ejecute como debería y prueban lo siguiente:

* Un programa que utilice estatutos condicionales, es decir, if, else, elsif y unless.
* Un programa que utilice estatutos de repetición, while y do while.
* Un programa que utilice funciones dentro de la clase main.
* Un programa con clases.
* Un programa que utilice funciones con recursividad.
* Un programa donde se use el resultado de una función como parámetro de otra función.
* Programa con un error de sintaxis para que el compilador marque la línea del error y el tipo de error.
* Programa que cuente con una división entre 0.
* Un programa que implemente herencia entre clases.
* Un programa que ordene elementos dentro de un arreglo.
* Un programa con multiplicación de matrices.
* Un programa con un error de semántica.

## A.4) Descripción del proceso de desarrollo

El proyecto fue dividido en diferentes etapas, de esta forma se buscó desarrollar todo el proyecto por partes de un peso relativamente equivalente. Estas se muestran a continuación junto con el avance real:

* Primera etapa: consiste en realizar el léxico y sintaxis del lenguaje. Avance real: Reporte de avance #1: Para esta entrega se tienen los archivos de Rex y Racc con el léxico y la gramática completos, es decir, ya comprenden todo el lenguaje. No obstante, el programa no funciona debido a que hay conflictos de tipo shift/reduce y reduce/reduce. Debido a la falta de tiempo la entrega se hace con algunos de estos errores aunque cabe mencionar que se están tratando de resolver lo más rápido posible para que no se acumulen con la siguiente entrega.
* Segunda etapa: diseñar el directorio de procedimientos y las tablas de variables. Avance real: Se cuenta con la teoría para crear los directorios de funciones y las tablas de variables. Actualmente se está trabajando en el código para poder implementar esto. A diferencia de la entrega pasada, el código ya funciona y no tiene conflictos de shift/reduce y reduce/reduce.
* Tercera etapa: diseñar el cubo semántico y poblarlo, también crear los cuádruplos para expresiones lineales. Avance real: Las entregas pasadas ya funcionan y de esta entrega ya se checa casi al 100 la semántica de variables, queda pendiente la semántica de expresiones.
* Cuarta etapa: generar los cuádruplos para los demás estatutos. Avance real: El proyecto ya cuenta con un Directorio de Procedimientos y tablas de variables. Estos ya se llenan con datos pero aún falta agregar las validaciones del cubo y todo lo que concierne a los estatutos. No obstante, pensamos terminar con lo anterior para este fin de semana. Se esperaría que para el lunes ya estuviera funcionando todo el cubo y que se guarden todos los cuádruplos para todos los estatutos.
* Quinta etapa: diseñar todos los cuádruplos y semántica necesaria para funciones. Avance real: El proyecto ya cumple con casi todo lo que se ocupaba hasta esta entrega, lo único que hace falta es agregar la generación de cuádruplos para las llamadas a funciones. Fuera de eso, ya está toda la validación semántica y generación de cuádruplos para estatutos y funciones.
* Sexta etapa: crear el mapa de memoria y la máquina virtual funcionando para lo que se tiene hasta el momento. Avance real: Ya se creó la máquina virtual y funcionan los estatus simples (Aquellos que no involucran saltos o funciones). En cuanto a la parte de compilación, lo único que falta agregar son los arreglos. Falta hacer unos pequeños cambios en la máquina virtual para que se realicen las operaciones de asignación con diferentes operadores (+=, -=, etc.).
* Séptima etapa: agregar los arreglos, tomar sus consideraciones semánticas en cuenta, crear sus cuádruplos y agregarlos a la máquina virtual. Avance real: Ya se cumple con un avance del 95% aproximadamente. Se agregaron arreglos (incluso se agregaron arreglos multidimensionales). Ya funcionan pruebas con arreglos, recursividad, clases y una combinación de todo. Falta arreglar un detalle con los números negativos ya que estos no son captados, no por el lenguaje, sino por la herramienta que estamos utilizando. Fuera de eso faltaría terminar la documentación.
* Octava etapa: agregar funcionalidad extra y avanzarle a la documentación. Avance real: Ya están tanto el compilador como la máquina virtual con la funcionalidad original y la funcionalidad extra. Se anexa la primera versión de la documentación. Falta hacer unos cambios y agregar los diagramas de sintaxis con las consideraciones semánticas. También falta agregar el manual de usuario y el video que ya está hecho, solo que ese se va a poner en el USB de la entrega final.

### Reflexión José Manuel:

Personalmente me gustó mucho este proyecto, más que nada por el reto que implicó y por los aprendizaje que obtuve gracias a realizarlo. Considero que el proyecto tuvo partes más complicadas que otras, sobre todo a la mitad de todo el desarrollo cuando nos dimos cuenta que íbamos un tanto atrasados. A partir de ese momento aumentó mi preocupación y por lo mismo el nivel de dedicación que le di al proyecto. Es cierto que hubo partes que no pude hacer más que nada por el tiempo que tenía, no obstante una vez que mi compañero implementó esas cosas (funciones, arreglos multidimensionales), me tomaba mi tiempo para entender cómo funcionaban estos. Además, me gusta el grado al cual aprendí a utilizar mis conocimientos de otras materias, sobre todo estructuras de datos. Creo que nunca había utilizado tantas estructuras en un proyecto, sobre todo muchas estructuras muy complejas como hashes dentro de hashes y así sucesivamente. Por otro lado creo que aumentó el nivel de aprendizaje que tenía sobre Ruby ya que me consideraba como alguien con un nivel básico pero ahora puedo leer el código de otras gemas y puedo entender qué es lo que hizo el desarrollador original, espero que pase lo mismo cuando alguien más lea el código de este proyecto. En general considero que fue una experiencia muy interesante y enriquecedora. Este es definitivamente el proyecto del cual estoy más orgulloso en lo que llevo de toda mi carrera y tengo el presentimiento de que así será hasta que me gradúe.

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

José Manuel González Castro

### Reflexión Axel Suárez:

Para mí este proyecto representó la oportunidad de aplicar múltiples áreas de conocimiento desarrolladas durante la carrera en un programa con cierto nivel de dificultad que puede ser utilizado gracias a su funcionalidad. Aunque al principio pensamos que un lenguaje orientado a objetos no debía conllevar una dificultad muy elevada por sobre uno procedural, al darnos cuenta que el manejo de los mismo volvían casi todos los aspectos de la compilación más complejos, esto nos motivó a hacer un proyecto completo que tuviera la funcionalidad deseada. En mi caso que trabajé en la parte de funciones, manejo de objetos y variables multidimensionales puedo asegurar que aprendí demasiadas cosas con el curso, desde el manejo de hashes imbricados en Ruby, hasta el manejo de procedimientos recursivos a través de un delicado manejo de pilas. Me gustó mucho el pensar cómo se manejaría el concepto de un objeto que mandaba a llamar un método de su clase y que necesitaba modificar sus atributos relacionados en el contexto actual, y el poder resolver esta abstracción me hace apreciar a los lenguajes orientados a objetos mucho más. Por otro lado, otro punto que me deja una enorme satisfacción es el haber utilizado tecnologías poco comunes para este proyecto como lo es Ruby, y aún más las herramientas de análisis léxico y sintáctico: Rex y Racc. El hecho de que ambas herramientas tuvieran una pobre documentación y pocos ejemplos me hizo desarrollar la habilidad de entender la implementación de herramientas encontradas en internet, y el poder usar muchas funcionalidades escondidas de Racc también me facilitó mucho trabajo.

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

Raúl Axel Suárez Martínez

# Descripción del Lenguaje

A continuación se habla un poco sobre el lenguaje; su nombre, sus principales características y los posibles errores que pueden surgir durante la compilación y la ejecución de este.

## B.1) Nombre del lenguaje

El nombre del lenguaje es Objective Plox

## B.2) Descripción de las principales características del lenguaje

Ya se mencionaron algunas de las características que va a tener el lenguaje. Para mostrarlas de una manera más sencilla se tiene el siguiente listado.

* Lenguaje orientado a objetos.
* Permite herencia entre clases.
* Permite declarar arreglos multidimensionales.
* Cuenta con características básicas de un lenguaje orientado a objetos; múltiples clases (en este caso todas en el mismo archivo), estatutos de repetición, estatutos de decisión, operaciones aritméticas, operaciones de asignación, impresiones, capacidad de tener input del teclado.
* Cuenta con funciones públicas y privadas dentro de clases.
* Funciones con valor de retorno de todos los tipos de datos del lenguaje.
* Funciones recursivas.
* Uso del resultado de una función como parámetro de otra función.
* Palabras reservadas con nombres llamativos.

## B.3) Descripción de los posibles errores

Durante compilación y ejecución pueden ocurrir diferentes, estos se muestran a continuación:

**Compilación**:

* Error si el nombre de la clase ya existe en el directorio de clases.
* Error si el nombre de la clase de la cual se hereda no se encuentra en el directorio de clases.
* Error si se está tratando de acceder a una clase que no existe todavía dentro del directorio de clases.
* Error si se está tratando de crear una clase dentro de otra clase. Solo se permiten clases por si solas.
* Error si se está creando una variable con el nombre de otra variable que ya existe dentro del contexto actual del programa.
* Error si el nombre de una variable ya existe en otro método donde se está utilizando como variable pública o global.
* Error al tratar de crear un arreglo con tipo de dato de un objeto creado.
* Error al tener más de un argumento con el mismo nombre dentro de una función.
* Error al tratar de aplicar operadores no válidos entre diferentes tipos de datos.
* Error al tratar de aplicar la negación con tipos de datos diferentes a logic.
* Error al tratar de utilizar el signo de – entre dos tipos de dato no numéricos.
* Error cuando se trata de utilizar una variable que aun no ha sido declarada y no se encuentra dentro de la tabla de variables.
* Error cuando se trata de utilizar un arreglo y este no ha sido declarado.
* Error al tener un valor de retorno dentro de una función que no tiene valor de retorno.
* Error al tratar de acceder a un método que no es público dentro de otro contexto.
* Error al tratar de utilizar un método que aún no ha sido creado.
* Error al tratar de utilizar una función mandando una cantidad diferente de argumentos a los que en realidad debería tener.
* Error al tratar de acceder a un atributo que no ha sido declarado como público en otro contexto.
* Error si se trata de acceder al índice de un arreglo con algo diferente a un número.
* Error si se ponen más o menos dimensiones de las que se tienen en un arreglo.

**Máquina virtual**:

* Error al realizar una división entre 0.
* Error al tratar de acceder a una casilla inexistente dentro de un arreglo.

Cabe destacar que todos los errores muestran un mensaje apropiado, indicando por qué tronó el compilador o la máquina virtual. Además, se muestra la línea en la que ocurrió el error. Esto solo sucede una vez puesto que el proceso termina cuando se encuentra el primer error.

# Descripción del Compilador

En este apartado se describen todos los componentes del compilador, esto quiere decir que se explica en qué fue desarrollado, el léxico, sintaxis y semántica de este al igual que la forma en la que se guarda toda la información, tanto los cuádruplos como la memoria.

## C.1) Equipo de cómputo, lenguaje y utilerías usadas en el proyecto

El compilador está hecho en Ruby y dado que no es tan fácil trabajar con Ruby en una computadora con sistema operativo Windows, los equipos de cómputo utilizados fueron Macs (Mac mini y Macbook Air). En este caso las utilerías para crear el compilador fueron Rex y Racc, herramientas equivalentes a Flex y Bison o Lex y Yacc solo que en este caso funciona para Ruby. Rex es la herramienta equivalente al scanner mientras que Racc es el parser. Otras utilerías que se utilizaron fueron awesome print para poder imprimir las estructuras con estilo para que fuera más fácil ver si todo se guardaba bien y lexer para saber en qué línea se encuentran los errores.

## C.2) Descripción del Análisis Léxico

A continuación se muestra el léxico del lenguaje, es decir, tanto los elementos principales como los tokens que tiene el lenguaje e incluso las palabras reservadas del lenguaje.

### C.2.1) Patrones de construcción de los elementos principales

El lenguaje cuenta con 4 elementos principales, estos son números enteros, números decimales, identificadores y constantes string, que también sirve para los caracteres. Para ilustrar estos elementos se muestran sus patrones de construcción.

Números enteros: \d+

Números flotantes: \d+\.\d+

Identificadores: [a-zA-Z][a-zA-Z0-9]\*

Constantes string: ".\*"

### C.2.2) Tokens del lenguaje y código asociado

Al igual que con los patrones de construcción, a continuación se muestran los tokens del lenguaje junto al código equivalente para poder implementarlos.

|  |  |
| --- | --- |
| Token | Código asociado |
| != | != { [:DIFFERENT, {0 => text}] } |
| ||= | \|\|= { [:ORASSIGN, {0 => text}] } |
| &&= | &&= { [:ANDASSIGN, {0 => text}] } |
| += | \+= { [:PLUSASSIGN, {0 => text}] } |
| -= | \-= { [:MINUSASSIGN, {0 => text}] } |
| \*= | \\*= { [:MULTASSIGN, {0 => text}] } |
| /= | \/= { [:DIVASSIGN, {0 => text}] } |
| %= | %= { [:MODASSIGN, {0 => text}] } |
| == | == { [:EQUALITY, {0 => text}] } |
| <= | \<= { [:LEQUAL, {0 => text}] } |
| >= | \>= { [:MEQUAL, {0 => text}] } |
| && | && { [:AND, {0 => text}] } |
| || | \|\| { [:OR, {0 => text}] } |
| ! | ! { [:NOT, {0 => text}] } |
| % | % { [:MOD, {0 => text}] } |
| \* | \\* { [:MULT, {0 => text}]} |
| / | \/ { [:DIV, {0 => text}]} |
| + | \+ { [:PLUS, {0 => text}]} |
| - | \- { [:MINUS, {0 => text}] } |
| ( | \( { [:PLEFT, {0 => text}] } |
| ) | \) { [:PRIGHT, {0 => text}] } |
| { | \{ { [:BLEFT, {0 => text}] } |
| } | \} { [:BRIGHT, {0 => text}] } |
| [ | \[ { [:SBLEFT, {0 => text}] } |
| ] | \] { [:SBRIGHT, {0 => text}] } |
| : | : { [:TWOP, {0 => text}] } |
| ; | ; { [:SEMIC, {0 => text}] } |
| = | = { [:EQUAL, {0 => text}] } |
| , | , { [:COMA, {0 => text}] } |
| . | \. { [:POINT, {0 => text}] } |
| < | \< { [:LTHAN, {0 => text}] } |
| > | \> { [:MTHAN, {0 => text}] } |

C.2.3) Palabras reservadas

La siguiente tabla ilustra las palabras reservadas del lenguaje así como su equivalente en otro lenguaje. Esto se ilustra como (palabra reservada en otro lenguaje, nombre del lenguaje).

|  |  |
| --- | --- |
| Palabra reservada | Palabra reservada equivalente y lenguaje |
| habemvs | package, Java |
| oblivion | void, Java |
| var | var, Javascript |
| eternal | const, Java |
| chief | main, Java |
| hidden | private, Java |
| open | public, Java |
| species | class, Java |
| null | null, Java |
| if | if, Java |
| else | else, Java |
| elsif | elsif, Ruby |
| while | while, Java |
| do | do, Java |
| unless | unless, Ruby |
| true | true, Java |
| false | false, Java |
| say | printf, C++ |
| hear | scanf, C++ |
| funk | function, Javascript |
| heirof | extends, Java |
| reply | return, Java |
| number | int, Java |
| decimal | float, Java |
| char | char, Java |
| string | String, Java |
| logic | boolean, Java |

## C.3) Descripción del Análisis de Sintaxis

Para crear la gramática del lenguaje se utilizó la notación EBNF más que nada porque la herramienta que permite crear los diagramas de sintaxis utiliza dicha notación. En el siguiente punto se muestra toda la gramática con EBNF.

### C.3.1) Gramática Formal

plox\_generation ::=

(type\_declaration)\*

type\_declaration ::=

"habemvs" class\_declaration

class\_declaration ::=

"species" identifier

( "heirof" identifier )?

"{" (variable\_declaration)\* (method\_declaration)\* "}"

modifier ::=

"open"

| "hidden"

variable\_declaration ::=

( "var" | "eternal" ) ( modifier )? type\_specifier identifier (array\_dec)?

( "," identifier (array\_dec)? )\* ";"

array\_dec ::=

"[" CTEN ".." CTEN ("," CTEN ".." CTEN)\* "]"

method\_declaration ::=

"funk" ( modifier )? type ( identifier | "chief" )

"(" ( parameter\_list )\* ")"

( statement\_block | ";" )

statement\_block ::= "{" (variable\_declaration)\* ( statement )\* "}"

variable\_assignment ::=

identifier ( "[" expression "]" )\* assign\_operator expression

assign\_operator ::=

"="

| "+="

| "-="

| "\*="

| "/="

| "||="

| "&&="

| "%="

parameter\_list ::=

parameter ( "," parameter\_list )\*

parameter ::=

type identifier

statement ::=

variable\_assignment ";"

| ( "hear" "(" expression ")" ";")

| ( "say" "(" expression ")" ";")

| ( unless\_statement )

| ( if\_statement )

| ( do\_statement )

| ( while\_statement )

| ( "reply" ( expression )? ";" )

| ( function\_call ";" )

| ( identifier "." function\_call ";" )

unless\_statement ::=

"unless" "(" expression ")" statement

( "else" statement )?

if\_statement ::=

"if" "(" expression ")" statement

( "else" statement )?

do\_statement ::=

"do" statement "while" "(" expression ")" ";"

while\_statement ::=

"while" "(" expression ")" statement

expression ::=

( expression num\_operator expression )

| ( expression testing\_operator expression )

| ( "!" expression )

| ( expression boolean\_operator expression )

| ( "true" )

| ( "false" )

| ( "(" expression ")" )

| ( literal\_expression )

| ( reference\_expression )

reference\_expression ::=

identifier

| reference\_expression ( ( "(" ( arglist )? ")" )

| ( "[" expression "]" )

| ( "." reference\_expression )

)

literal\_expression ::=

cted

| cten

| ctestring

arglist ::=

expression ( "," arglist )\*

type\_specifier ::=

"logic"

| "char"

| "number"

| "decimal"

| identifier

| "oblivion"

| "string"

num\_operator ::=

"\*"

| "/"

| "%"

| "+"

| "-"

testing\_operator ::=

">"

| "<"

| ">="

| "<="

| "=="

| "!="

boolean\_operator ::=

"&&"

| "||"

## C.4) Descripción de Generación de Código Intermedio y Análisis Semántico

En lo que corresponde a la parte semántica del lenguaje, se tuvieron que poner reglas entre los elementos sintácticos para poder generar el código intermedio correspondiente. Para esto se utilizaron diferentes estructuras mencionadas en los siguientes puntos.

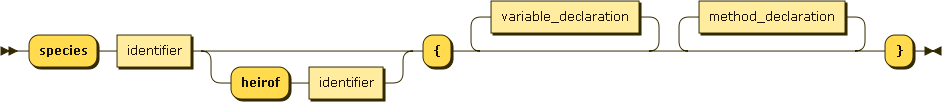
### C.4.1) Código de operación y direcciones virtuales asociadas

La representación que decidimos utilizar para la generación de código intermedio fueron cuádruplos. El lenguaje cuenta con diferentes tipos de cuádruplos para cada una de las acciones que se deseen realizar, estos se pueden encontrar en el apartado C.5.1. En cuanto a las direcciones virtuales, se utilizó una estructura de datos que corresponde a un hash de hashes con 2 niveles. En el primer nivel del hash se encuentran los diferentes contextos del lenguaje; global, local, temporal y constantes. Cada uno de estos contextos tiene su propio hash interno con los distintos tipos de datos. Estos tipos de datos indican un número entre 0 y 19 dependiendo del contexto actual. Además se tiene otro hash con la misma estructura que guarda la cantidad de variables o elementos que han sido guardados en la memoria. Estas estructuras se pueden encontrar en el apartado C.5.1 al igual que los cuádruplos.

### C.4.2) Diagramas de sintaxis con las acciones correspondientes

En esta parte se presentan los diagramas de sintaxis junto con las reglas correspondientes indicando qué se hace en las partes señaladas. Algunos no tienen reglas porque no se realiza nada ahí.

Nueva Clase:

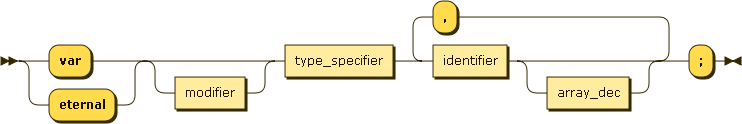


1

2

1. Se checa si la clase no existe previamente, sino se da de alta en el directorio de clases. La variable actualSpecies = identifier y actualMethod = “species” porque se empieza en el ambiente de clase (global).
2. Checa que dicho padre exista, si es correcto se pone un apuntador a la clase padre en la llave “father” y se copia el tamaño de la clase padre en la clase actual.

Nueva variable:



1

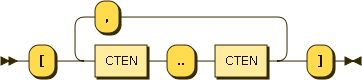
2

3

4

1. isVariable = true/false
2. actualModifier = true/false
3. actualType = type\_specifier
4. Dar de alta la variable en el directorio de procedimientos según sea el actualSpecies como atributo de clase o variable de función. Guardar el tipo, el modifier y si es variable de las variables anteriores. Checar si el tipo de la variable es valido: si no es de tipo primitivo entonces checar en la tabla de variable si la clase esta definida. Si es de tipo objeto, se parte la variable según todos los atributos de la clase y de las clases padre. Se saca una dirección y se guarda.

Declaracion de arreglo:



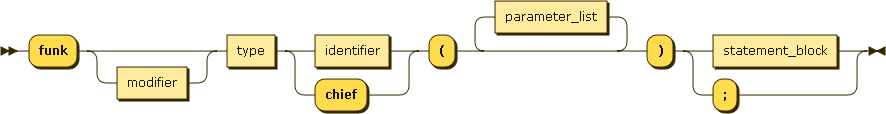
2

3

1

1. Se checa que el tipo del objeto actual (actualType) es primitivo porque no se soportan arreglos de objetos. Se empieza a crear la “lista” de dimensiones. Se indica que es una variable dimensionada en el directorio de clases
2. Se hace un hash donde se guardan los limites superior e inferior, así como la “r”: (limite sup – lim inf + 1) \* r. Éste hash se agrega a la lista de dimensiones.
3. Con la ultima “r” se hace el cálculo de las “m” y las “k”. Se separa el espacio de memoria completo que requiere la variable.

Declaración de un método:



2

1

3

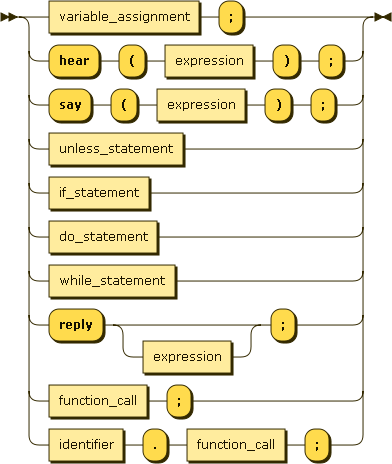
4

5

6

1. actualModifier = true (porque las funciones son por default públicas).
2. ActualModifier = modifier
3. actualType = type
4. Se da de alta la función en la tabla de clases/procedimientos, se valida que no exista otra función con el mismo nombre en la clase actual y en caso de que la función sea chief, se llena el cuádruplo de goto inicial para que sepa donde empezar el programa. Se checa que el tipo de la variable sea un tipo primitivo y se dan de alta los datos de modifier, type y el cuádruplo donde va a empezar la función. Se reinician los contadores locales.
5. Por cada tipo y id de parámetro: se checa que el tipo de parámetro sea valido, se da de alta el nombre en la tabla de variables de la función y se checa que no existe previamente una variable con el mismo nombre. Se saca la dirección del dato y se agrega junto con el tipo a la lista de argumentos de la función.
6. Se genera el cuádruplo de retorno de la función.

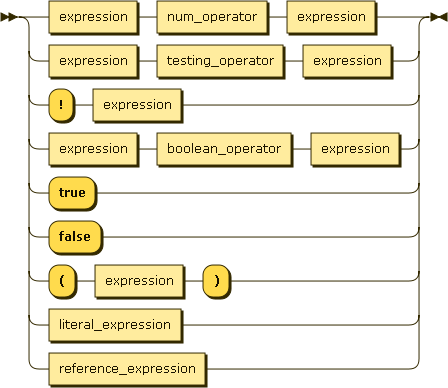
Expresiones:



1

1. Se genera cuádruplo con el valor de la expresión

Expresión:



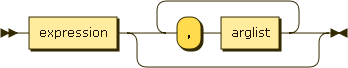
1

\*Sintácticamente al reducir la regla ya se tienen los valores y sus tipos, por lo que no hay una pila de operadores y/u operandos.

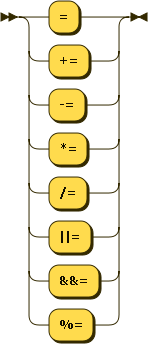
1. Se checa el tipo de ambas expresiones y se checa en el cubo que sea una operación válida. Se genera el cuádruplo con las dos direcciones de memoria

Resto de los diagramas:

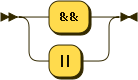
Arglist:



Assign operator:



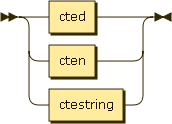
Boolean operator:



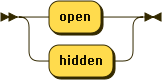
Do statement:

../../../../Downloads/diagram-6/diagram/do_statement.png

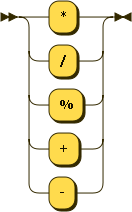
If statement:



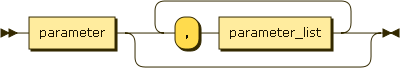
Modifier:



Num operator:



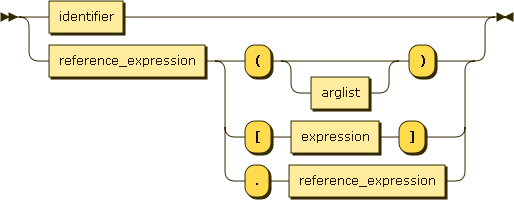
Parameter list:



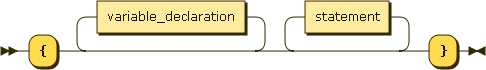
Parameter:

../../../../Downloads/diagram-6/diagram/parameter.png

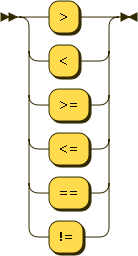
Reference expression:



Statement block:



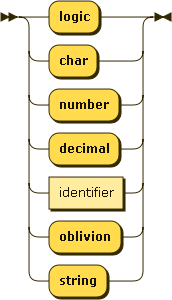
Testign operator

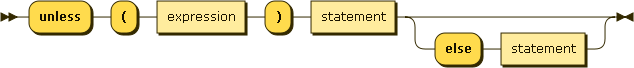


Type declaration:

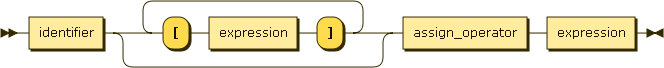
../../../../Downloads/diagram-6/diagram/type_declaration.png

Type specifier:



Unless statement:

Variable Assignment:



While statement:

../../../../Downloads/diagram-6/diagram/while_statement.png

### C.4.3) Descripción de las acciones semánticas y de código

El lenguaje cuenta con varias acciones semánticas como se pudo ver en el punto anterior. Algunas de estas quedaron un poco ambiguas por lo que van a ser descritas en el siguiente listado, indicando la acción semántica que se está haciendo y el error que puede ocurrir.

* Creando una clase: error si el nombre de la clase ya existe en el directorio de clases.
* Validando herencia: error si el nombre de la clase de la cual se hereda no se encuentra en el directorio de clases.
* Clase no declarada: error si se está tratando de acceder a una clase que no existe todavía dentro del directorio de clases.
* Clases dentro de clases: error si se está tratando de crear una clase dentro de otra clase. Solo se permiten clases por si solas.
* Variable definida anteriormente: error si se está creando una variable con el nombre de otra variable que ya existe dentro del contexto actual del programa.
* Variable definida en método: error si el nombre de una variable ya existe en otro método donde se está utilizando como variable pública o global.
* Arreglos de objetos: error al tratar de crear un arreglo con tipo de dato de un objeto creado.
* Argumentos de funciones: error al tener más de un argumento con el mismo nombre dentro de una función.
* Tipos de datos no compatibles: error al tratar de aplicar operadores no válidos entre diferentes tipos de datos. Esto se puede ver mejor con la explicación del cubo semántico en el punto C.4.4.
* Negación con tipos no lógicos: error al tratar de aplicar la negación con tipos de datos diferentes a logic.
* Signo de – con operadores no numéricos: error al tratar de utilizar el signo de – entre dos tipos de dato no numéricos.
* Variable no declarada: error cuando se trata de utilizar una variable que aun no ha sido declarada y no se encuentra dentro de la tabla de variables.
* Variable dimensionada no declarada: error cuando se trata de utilizar un arreglo y este no ha sido declarado.
* Retorno en funciones oblivion: error al tener un valor de retorno dentro de una función que no tiene valor de retorno.
* Acceso inválido a un método: error al tratar de acceder a un método que no es público dentro de otro contexto.
* Método no definido: error al tratar de utilizar un método que aun no ha sido creado.
* Cantidad de argumentos: error al tratar de utilizar una función mandando una cantidad diferente de argumentos a los que en realidad debería tener.
* Acceso inválido a atributos: error al tratar de acceder a un atributo que no ha sido declarado como público en otro contexto.
* Obtener índice de un arreglo: error si se trata de acceder al índice de un arreglo con algo diferente a un número.
* Diferente cantidad de dimensiones: error si se ponen más o menos dimensiones de las que se tienen en un arreglo.

### C.4.4) Tabla de consideraciones semánticas

La tabla de consideraciones semánticas que utilizamos se puede representar por medio de un cubo. En cuanto a programación, el cubo no es más que un hash de hashes. En el primer nivel del hash está el primer tipo de dato, en el segundo nivel están los demás tipos de datos con los cuales se va a comparar y en el último nivel del hash se encuentran los tokens del lenguaje con los cuales hay compatibilidad, las llaves, si es que hay, indican el tipo de dato resultante con el token correspondiente. Es importante ver que no todos los niveles más bajos del hash tienen tokens debido a que la ausencia de tokens indica que no existe compatibilidad entre los tipos que se tienen. Para ilustrar mejor todo esto se muestra el cubo semántico completo a continuación:

$semanticCube = {

"logic" => {

"logic" => {

"!=" => "logic",

"||=" => "logic",

"&&=" => "logic",

"==" => "logic",

"&&" => "logic",

"||" => "logic",

"=" => "logic"

},

"char" => {

},

"number" => {

},

"decimal" => {

},

"string" => {

},

"hear" => {

"!=" => "logic",

"||=" => "logic",

"&&=" => "logic",

"==" => "logic",

"&&" => "logic",

"||" => "logic",

"=" => "logic"

}

},

"char" => {

"logic" => {

},

"char" => {

"!=" => "logic",

"==" => "logic",

"<=" => "logic",

">=" => "logic",

"&&" => "logic",

"||" => "logic",

"=" => "char",

"<" => "logic",

">" => "logic"

},

"number" => {

},

"decimal" => {

},

"string" => {

},

"hear" => {

"!=" => "logic",

"==" => "logic",

"<=" => "logic",

">=" => "logic",

"&&" => "logic",

"||" => "logic",

"=" => "char",

"<" => "logic",

">" => "logic"

}

},

"number" => {

"logic" => {

},

"char" => {

},

"number" => {

"!=" => "logic",

"+=" => "number",

"-=" => "number",

"\*=" => "number",

"/=" => "number",

"%=" => "number",

"==" => "logic",

"<=" => "logic",

">=" => "logic",

"&&" => "logic",

"||" => "logic",

"%" => "number",

"\*" => "number",

"/" => "number",

"+" => "number",

"-" => "number",

"=" => "number",

"<" => "logic",

">" => "logic"

},

"decimal" => {

"!=" => "logic",

"+=" => "number",

"-=" => "number",

"\*=" => "number",

"/=" => "number",

"%=" => "number",

"==" => "logic",

"<=" => "logic",

">=" => "logic",

"&&" => "logic",

"||" => "logic",

"%" => "number",

"\*" => "number",

"/" => "number",

"+" => "number",

"-" => "number",

"=" => "number",

"<" => "logic",

">" => "logic"

},

"string" => {

},

"hear" => {

"!=" => "logic",

"+=" => "number",

"-=" => "number",

"\*=" => "number",

"/=" => "number",

"%=" => "number",

"==" => "logic",

"<=" => "logic",

">=" => "logic",

"&&" => "logic",

"||" => "logic",

"%" => "number",

"\*" => "number",

"/" => "number",

"+" => "number",

"-" => "number",

"=" => "number",

"<" => "logic",

">" => "logic"

}

},

"decimal" => {

"logic" => {

},

"char" => {

},

"number" => {

"!=" => "logic",

"+=" => "number",

"-=" => "number",

"\*=" => "number",

"/=" => "number",

"%=" => "number",

"==" => "logic",

"<=" => "logic",

">=" => "logic",

"&&" => "logic",

"||" => "logic",

"%" => "number",

"\*" => "number",

"/" => "number",

"+" => "number",

"-" => "number",

"=" => "number",

"<" => "logic",

">" => "logic"

},

"decimal" => {

"!=" => "logic",

"+=" => "decimal",

"-=" => "decimal",

"\*=" => "decimal",

"/=" => "decimal",

"%=" => "decimal",

"==" => "logic",

"<=" => "logic",

">=" => "logic",

"&&" => "logic",

"||" => "logic",

"%" => "decimal",

"\*" => "decimal",

"/" => "decimal",

"+" => "decimal",

"-" => "decimal",

"=" => "decimal",

"<" => "logic",

">" => "logic"

},

"string" => {

},

"hear" => {

"!=" => "logic",

"+=" => "decimal",

"-=" => "decimal",

"\*=" => "decimal",

"/=" => "decimal",

"%=" => "decimal",

"==" => "logic",

"<=" => "logic",

">=" => "logic",

"&&" => "logic",

"||" => "logic",

"%" => "decimal",

"\*" => "decimal",

"/" => "decimal",

"+" => "decimal",

"-" => "decimal",

"=" => "decimal",

"<" => "logic",

">" => "logic"

}

},

"string" => {

"logic" => {

},

"char" => {

},

"number" => {

},

"decimal" => {

},

"string" => {

"!=" => "logic",

"==" => "logic",

"&&" => "logic",

"||" => "logic",

"=" => "string",

},

"hear" => {

"!=" => "logic",

"==" => "logic",

"&&" => "logic",

"||" => "logic",

"=" => "string",

}

}

}

## C.5) Descripción detallada del proceso de Administración de Memoria

En este apartado se muestran las diferentes estructuras de datos que fueron utilizadas, esto cubre tanto la tabla de procedimientos como las tablas de variables, los cuádruplos que se utilizaron para la generación de código intermedio, el cubo semántico y las estructuras base para la generación de la memoria.

### C.5.1) Especificación gráfica de las estructuras de datos utilizadas

**Tabla de procedimientos/clases**: Hash de hashes que se puede representar de la siguiente manera

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre de la clase | Tabla de métodos | Tabla de variables | Tabla de tamaños | Padre |
|  |  |  |  |  |

* Nombre de la clase indica el nombre de la clase actual.
* Tabla de métodos es un apuntador a un hash con los métodos de la clase.
* Tabla de variables es un apuntador a un hash con las variables de la clase.
* Tabla de tamaños es un apuntador a un hash con la cantidad de variables de cada tipo dentro de la clase.
* Padre es un apuntador a los datos de la clase padre.

**Tabla de métodos:** Hash de hashes que se ve de la siguiente manera

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nombre del método | Tipo de dato | Scope | Tabla de tamaños | Tabla de variables | Tabla de argumentos | Inicio |
|  |  |  |  |  |  |  |

* Nombre del método indica el nombre del método actual.
* Tipo de dato indica el tipo si es que el método tiene un valor de retorno o un oblivion si es que no tiene.
* Scope se marca como verdadero por defecto e indica si el método es público o privado.
* Tabla de tamaños es un apuntador a un hash con la cantidad de variables de cada tipo dentro de la clase.
* Tabla de variables es un apuntador a un hash con las variables de la clase.
* Tabla de argumentos es un apuntador a un hash con los datos de los argumentos de la función.
* Inicio es el contador del método actual en el contexto actual.

**Tabla de variables:** Hash que se representa de la siguiente forma

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la variable | Tipo de dato | Modificable | Dirección |
|  |  |  |  |

* Nombre de la variable indica el nombre de la variable creada.
* Tipo de dato indica el tipo de dato de la variable.
* Modificable indica si la variable es pública o privada.
* Dirección es la dirección de memoria donde está guardada dicha variable.

**Tabla de tamaños:** Hash que se representa de la siguiente manera

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number | Decimal | String | Char | Logic |
|  |  |  |  |  |

* Number indica la cantidad de variables number que hay en el contexto al que pertenece la tabla.
* Decimal indica la cantidad de variables decimal que hay en el contexto al que pertenece la tabla.
* String indica la cantidad de variables string que hay en el contexto al que pertenece la tabla.
* Char indica la cantidad de variables char que hay en el contexto al que pertenece la tabla.
* Logic indica la cantidad de variables logic que hay en el contexto al que pertenece la tabla.

**Tabla de argumentos:**

|  |  |  |
| --- | --- | --- |
| Subíndice del argumento | Dirección | Tipo de dato |
|  |  |  |

* Subíndice del argumento indica el argumento actual.
* Dirección indica la dirección de memoria del argumento.
* Tipo de dato indica el tipo de dato al que corresponde el argumento.

**Cuádruplos:** La estructura que se usó para guardar los cuádruplos es una matriz donde la primera dimensión representa el contador del cuádruplo y la segunda tiene 4 arreglos indexados desde el 0 al 3 con los valores del cuádruplo. En sí, cada cuádruplo se ve como una lista de listas. Esto se puede ver de la siguiente manera:

[ 0] [

[0] "goto",

[1] nil,

[2] nil,

[3] 19

],

[ 1] [

[0] "return",

[1] nil,

[2] nil,

[3] nil

],

[ 2] [

[0] "return",

[1] nil,

[2] nil,

[3] nil

]

El lenguaje cuenta con diferentes acciones dentro de cada cuádruplo. La acción siempre se encuentra en el primer elemento de la lista de cada cuádruplo. Dependiendo de la acción los demás elementos pueden variar. A continuación se muestra una tabla con la estructura de todos los cuádruplos disponibles en el lenguaje.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Elemento #1 | Elemento #2 | Elemento #3 | Elemento #4 | Acción |
| + | Var1 | Var2 | Temp | Suma dos variables y las guarda en un temporal. |
| - | Var1 | Var2 | Temp | Resta dos variables y las guarda en un temporal. |
| \* | Var1 | Var2 | Temp | Multiplica dos variables y las guarda en un temporal. |
| / | Var1 | Var2 | Temp | Divide dos variables y las guarda en un temporal. |
| % | Var1 | Var2 | Temp | Obtiene el residuo entre dos variables y las guarda en un temporal. |
| > | Var1 | Var2 | Temp | Determina si una variable es mayor que otra y lo guarda en un temporal. |
| < | Var1 | Var2 | Temp | Determina si una variable es menor que otra y lo guarda en un temporal. |
| >= | Var1 | Var2 | Temp | Determina si una variable es mayor o igual que otra y lo guarda en un temporal. |
| <= | Var1 | Var2 | Temp | Determina si una variable es menor o igual que otra y lo guarda en un temporal. |
| == | Var1 | Var2 | Temp | Determina si dos variables son iguales y lo guarda en un temporal. |
| != | Var1 | Var2 | Temp | Determina si dos variables son diferentes y lo guarda en un temporal. |
| ! | - | Var1 | Temp | Negación de una variable que se guarda en un temporal. |
| && | Var1 | Var2 | Temp | Determina el equivalente a Var1 y Var2 y guarda el resultado en un temporal. |
| || | Var1 | Var2 | Temp | Determina el equivalente a Var1 o Var2 y guarda el resultado en un temporal. |
| = | - | Var1 | Temp | Asigna el valor de una variable a un temporal. |
| += | - | Var1 | Temp | Realiza una asignación con suma a Var1 y se guarda en un temporal. |
| -= | - | Var1 | Temp | Realiza una asignación con resta a Var1 y se guarda en un temporal. |
| \*= | - | Var1 | Temp | Realiza una asignación con multiplicación a Var1 y se guarda en un temporal. |
| /= | - | Var1 | Temp | Realiza una asignación con división a Var1 y se guarda en un temporal. |
| &&= | - | Var1 | Temp | Realiza una asignación con and a Var1 y se guarda en un temporal. |
| ||= | - | Var1 | Temp | Realiza una asignación con or a Var1 y se guarda en un temporal. |
| %= | - | Var1 | Temp | Realiza una asignación con módulo a Var1 y se guarda en un temporal. |
| negate | - | Var1 | Temp | Aplica el signo negativo a una variable y lo guarda en un temporal. |
| +SpecialRight | Var1 | Var2 | Temp | Suma una variable positiva con una negativa y lo guarda en un temporal. |
| \*SpecialRight | Var1 | Var2 | Temp | Multiplica una variable positiva con una negativa y lo guarda en un temporal. |
| verify | - | - | Var1 | Verifica que la variable esté entre los límites superior e inferior de un arreglo. |
| hear | Var1 | - | Temp | Captura una variable del teclado y lo guarda en un temporal. |
| say | Var1 | - | - | Imprime en pantalla el resultado de una variable. |
| goto | - | - | Dir | Hace un salto a una dirección. |
| gotoT | Res | - | Dir | Hace un salto a una dirección si el resultado es verdadero. |
| gotoF | Res | - | Dir | Hace un salto a una dirección si el resultado es falso. |
| param | Dir | Var1 | Dir | Dada una dirección con los parámetros, determina la cantidad y los manda a otra dirección. |
| gosub | Dir1 | - | Dir2 | Manda a la dirección de una función y guarda la dirección de regreso. |
| SEND\_ATTR | Var1 | - | Var2 | Manda un atributo a una función. |
| Return | - | - | Dir | Regresa a una dirección. |
| Terminate | - | - |  | Termina la compilación de un programa. |

**Cubo semántico:** Para poder ilustrar bien el cubo semántico se ha optado por crear diferentes tablas. Estas tablas tienen en la primera columna el tipo de dato principal a comparar. En la segunda columna se encuentran los demás tipos de dato contra los que se está comparando y en las siguientes columnas están los tokens. Las celdas indican el tipo de dato de retorno para la combinación de tipo de dato 1 con tipo de dato 2 y token. En caso de no haber match se indica con la palabra “Error”.

Tabla con tipo de dato char:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | !=, ==, <=, >=, &&, ||, <, > | = | ||=, &&=, +=, -=, \*=, /=, %=, !, %, \*, /, +, -,  (, ), [, ], {, }, :, ;, ‘,’ |
| char | **logic** | Error | Error | Error |
| char | **hear** | logic | char | Error |
| char | **number** | Error | Error | Error |
| char | **decimal** | Error | Error | Error |
| char | **string** | Error | Error | Error |
| char | **char** | logic | char | Error |

Tabla de tipo de dato logic:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | !=, ||, &&, ==, &&=, ||=, = | +=, -=, \*=, /=, %=, <=, >=, !, %, \*, /, +, -, (, ), {, }, [, ], :, ;, =, ‘,’, >, < |
| logic | **logic** | logic | Error |
| logic | **hear** | logic | Error |
| logic | **number** | Error | Error |
| logic | **decimal** | Error | Error |
| logic | **string** | Error | Error |
| logic | **char** | Error | Error |

Tabla de tipo de dato number:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | !=, ==, <=, >=, &&, ||, <, > | +=, -=, \*=, /=, %=, %, \*, /, +, -, = | ||=, &&=, !, (, ), [, ], {, }, :, ;, ‘,’ |
| number | **logic** | Error | Error | Error |
| number | **hear** | logic | number | Error |
| number | **number** | logic | number | Error |
| number | **decimal** | logic | number | Error |
| number | **string** | Error | Error | Error |
| number | **char** | Error | Error | Error |

Tabla de tipo de dato decimal:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | !=, ==, <=, >=, &&, ||, <, > | +=, -=, \*=, /=, %=, %, \*, /, +, -, = | ||=, &&=, !, (, ), [, ], {, }, :, ;, ‘,’ |
| decimal | **logic** | Error | Error | Error |
| decimal | **hear** | logic | decimals | Error |
| decimal | **number** | logic | number | Error |
| decimal | **decimal** | logic | decimal | Error |
| decimal | **string** | Error | Error | Error |
| decimal | **char** | Error | Error | Error |

Tabla de tipo de dato string:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | !=, ==, =, &&, || | = | ||=, &&=, !, (, ), [, ], {, }, :, ;, ‘,’, +=, -=, \*=, /=, %=, %, \*, /, +, -, <=, >=, <, > |
| string | **logic** | Error | Error | Error |
| string | **hear** | logic | string | Error |
| string | **number** | Error | Error | Error |
| string | **decimal** | Error | Error | Error |
| string | **string** | logic | string | Error |
| string | **char** | Error | Error | Error |

**Memoria:** Ya se mencionó brevemente pero en compilación hay un manejo de memoria que te permite saber el contexto actual, ya sea global, local, temporal o de constante. Esto se representa por un hash de hashes. La llave más interior es un número que permite identificar el contexto (del 0 al 19). Por otro lado hay una estructura igual donde la llave indica la cantidad de elementos del tipo dentro del contexto. Estas estructuras están dadas de la siguiente forma.

$magicReference = {

"global" => {

"number" => 0,

"decimal" => 1,

"string" => 2,

"char" => 3,

"logic" => 4

},

"local" => {

"number" => 5,

"decimal" => 6,

"string" => 7,

"char" => 8,

"logic" => 9

},

"temporal" => {

"number" => 10,

"decimal" => 11,

"string" => 12,

"char" => 13,

"logic" => 14

},

"constant" => {

"number" => 15,

"decimal" => 16,

"string" => 17,

"char" => 18,

"logic" => 19

}

}

$magicCounter = {

"global" => {

"number" => 0,

"decimal" => 0,

"string" => 0,

"char" => 0,

"logic" => 0

},

"local" => {

"number" => 0,

"decimal" => 0,

"string" => 0,

"char" => 0,

"logic" => 0

},

"temporal" => {

"number" => 0,

"decimal" => 0,

"string" => 0,

"char" => 0,

"logic" => 0

},

"constant" => {

"number" => 0,

"decimal" => 0,

"string" => 0,

"char" => 0,

"logic" => 0

}

}

# D) Descripción de la Máquina Virtual

En este apartado se habla sobre el proceso de creación de la máquina virtual, las herramientas que se utilizaron, la forma en que se manejó la memoria, la manera en la cual se recuperó la información de compilación, las estructuras utilizadas y ejemplos del código que se utilizó.

## D.1) Equipo de cómputo, lenguaje y utilerías especiales utilizadas

Al igual que con el compilador, el lenguaje utilizado como base fue Ruby. Por eso mismo, el equipo de cómputo utilizado fue el mismo, computadoras con Mac OSX (Mac mini y Macbook Air). En este caso no hubo necesidad de tener utilerías ya que la máquina virtual está construida con Ruby puro. La única utilería que tenemos en la máquina virtual permite imprimir los cuádruplos y las constantes en un archivo separado con extensión .yaml.

## D.2) Arquitectura del proceso de Administración de Memoria en ejecución

La máquina virtual cuenta con dos clases diferentes, una llamada memoria y una llamada máquina virtual. La clase máquina virtual es la que se encarga de inicializar todas las variables y de utilizar a la memoria. Después de compilación se manda a llamar la máquina virtual con un método que recorre los cuádruplos. Se obtiene el primer elemento dentro de cada cuádruplo (la acción) y se mapea con un método que realiza la acción correspondiente, obteniendo direcciones de la memoria y guardando los resultados cuando es necesario. En los siguientes puntos se va a hacer énfasis en cómo funciona la memoria y más adelante se van a mostrar ejemplos de código con las acciones de los cuádruplos y los accesos de memoria que se hacen.

### D.2.1) Especificación gráfica de las estructuras de datos utilizadas

### D.2.2) Asociación hecha entre las direcciones virtuales y reales

### D.2.3) Código con ejemplos de acceso a memoria

# E) Pruebas del funcionamiento del Lenguaje

Para probar el lenguaje completo se desarrollaron pruebas que cumplen con todo lo establecido en el punto A.3. En esta sección se muestran tanto el código de la prueba como el resultado de la misma.

## E.1) Pruebas que comprueban el funcionamiento del proyecto

Las pruebas a realizar corresponden a los test cases mencionados anteriormente. Para esto se crearon archivos que cumplen con la validación necesaria para ciertas pruebas. Esto se indica con el número de las pruebas en negritas seguido del código en el lenguaje. Las pruebas numeradas son las siguientes:

1. Un programa que utilice estatutos condicionales, es decir, if, else, elsif y unless.
2. Un programa que utilice estatutos de repetición, while y do while.
3. Un programa que utilice funciones dentro de la clase main.
4. Un programa con clases.
5. Un programa que utilice funciones con recursividad.
6. Un programa donde se use el resultado de una función como parámetro de otra función.
7. Programa con un error de sintaxis para que el compilador marque la línea del error y el tipo de error.
8. Programa que cuente con una división entre 0.
9. Un programa que implemente herencia entre clases.
10. Un programa que ordene elementos dentro de un arreglo.
11. Un programa con multiplicación de matrices.
12. Un programa con un error de semántica.

#### **Prueba #1:** Un programa que utilice estatutos condicionales, es decir, if, else, elsif y unless.

#### **Prueba #2:** Un programa que utilice estatutos de repetición, while y do while.

*Código:*

habemvs species EjemploObjectivePlox {

var hidden number a;

var open number b, c;

funk open oblivion chief() {

a = 50;

b = 10;

c = 35;

say("El valor de a es: ");

say(a);

if ( a > b || true ) {

say("Estoy en el if");

b = a + c;

while (a + b > 1) {

b += c;

c -= 10;

}

b += a;

}

elsif(a+b < c){

say("Estoy en el primer elsif");

a = b + c;

do{

a = a-1;

}while(a > b+c)

}

elsif(b == c){

say("Estoy en el segundo elsif");

say(a - c / 4);

}

else{

say("Estoy en el else");

a = a-1;

}

a = 50;

say("El valor final de a es: ");

say(a);

say("El valor final de b es: ");

say(b);

say("El valor final de c es: ");

say(c);

}

}

*Resultado del código intermedio:*

---

- - goto

-

-

- 1

- - =

-

- 150000

- 0

- - =

-

- 150001

- 1

- - =

-

- 150002

- 2

- - say

- 170000

-

-

- - say

- 0

-

-

- - '>'

- 0

- 1

- 140000

- - '||'

- 140000

- 190001

- 140001

- - gotoF

- 140001

-

- 20

- - say

- 170001

-

-

- - +

- 0

- 2

- 100000

- - =

-

- 100000

- 1

- - +

- 0

- 1

- 100001

- - '>'

- 100001

- 150003

- 140002

- - gotoF

- 140002

-

- 18

- - +=

-

- 2

- 1

- - -=

-

- 150001

- 2

- - goto

-

-

- 12

- - +=

-

- 0

- 1

- - goto

-

-

- 42

- - +

- 0

- 1

- 100002

- - <

- 100002

- 2

- 140003

- - gotoF

- 140003

-

- 32

- - say

- 170002

-

-

- - +

- 1

- 2

- 100003

- - =

-

- 100003

- 0

- - '-'

- 0

- 150003

- 100004

- - =

-

- 100004

- 0

- - +

- 1

- 2

- 100005

- - '>'

- 0

- 100005

- 140004

- - gotoV

- 140004

-

- 26

- - goto

-

-

- 42

- - ==

- 1

- 2

- 140005

- - gotoF

- 140005

-

- 39

- - say

- 170003

-

-

- - /

- 2

- 150004

- 100006

- - '-'

- 0

- 100006

- 100007

- - say

- 100007

-

-

- - goto

-

-

- 42

- - say

- 170004

-

-

- - '-'

- 0

- 150003

- 100008

- - =

-

- 100008

- 0

- - =

-

- 150000

- 0

- - say

- 170005

-

-

- - say

- 0

-

-

- - say

- 170006

-

-

- - say

- 1

-

-

- - say

- 170007

-

-

- - say

- 2

-

-

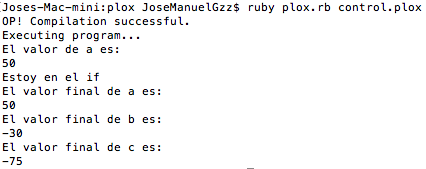
- - terminate

-

-

-

*Resultado del programa*:



#### **Prueba #3:** Un programa que utilice funciones dentro de la clase main.

*Código*:

habemvs species EjemploObjectivePlox {

var open number a;

funk number dos(number x) {

var number y;

y = x + 5;

reply y;

}

funk oblivion uno(number x) {

a = dos(x);

}

funk open oblivion chief() {

a = 50;

uno(15);

say("El resultado de a es: ");

say(a);

}

}

*Resultado de código intermedio*:

---

- - goto

-

-

- 12

- - +

- 50000

- 150000

- 100000

- - =

-

- 100000

- 50001

- - return

-

-

- 50001

- - return

-

-

-

- - param

- 50000

- - 1

- 1

- 50000

- - SEND\_ATTR

- 0

-

- 0

- - SEND\_ATTR

- 1

-

- 1

- - SEND\_ATTR

- 2

-

- 2

- - gosub

- 1

-

- 50001

- - =

-

- 50001

- 0

- - return

-

-

-

- - =

-

- 150001

- 0

- - =

-

- 150002

- 1

- - =

-

- 150003

- 2

- - param

- 150004

- - 1

- 1

- 50000

- - SEND\_ATTR

- 0

-

- 0

- - SEND\_ATTR

- 1

-

- 1

- - SEND\_ATTR

- 2

-

- 2

- - gosub

- 5

-

-

- - say

- 170000

-

-

- - say

- 0

-

-

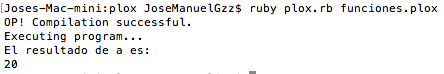
- - terminate

-

-

-

*Resultado del programa*:



#### **Prueba #4:** Un programa con clases.

#### **Prueba #5:** Un programa que utilice funciones con recursividad.

#### **Prueba #6:** Un programa donde se use el resultado de una función como parámetro de otra función.

#### **Prueba #9:** Un programa que implemente herencia entre clases.

*Código:*

habemvs species Funciones{

var number arr[2..4,0..2,1..6], resultado;

funk number fibonacci(number n){

var number f1, f2, aux, ciclo;

f1 = 0;

f2 = 1;

ciclo = 1;

while(ciclo < n){

aux = f1 + f2;

f1 = f2;

f2 = aux;

ciclo += 1;

}

say("Fibonacci iterativo:");

reply f2;

}

}

habemvs species FuncionesRecursivas heirof Funciones{

funk number fibonacci(number n){

if(n < 2){

reply n;

}

else{

reply fibonacci(n-1) + fibonacci(n-2);

}

}

}

habemvs species Tranqui {

funk oblivion chief(){

var Funciones prueba;

var FuncionesRecursivas pruebaRecursiva;

var number input;

say("Ingresa un numero");

hear(input);

say(prueba.fibonacci(prueba.fibonacci(input)));

say("Fibonacci recursivo:");

say(pruebaRecursiva.fibonacci(input));

say("OP");

}

}

*Resultado de código intermedio:*

---

- - goto

-

-

- 138

- - =

-

- 150000

- 50001

- - =

-

- 150001

- 50002

- - =

-

- 150001

- 50004

- - <

- 50004

- 50000

- 140000

- - gotoF

- 140000

-

- 12

- - +

- 50001

- 50002

- 100000

- - =

-

- 100000

- 50003

- - =

-

- 50002

- 50001

- - =

-

- 50003

- 50002

- - +=

-

- 150001

- 50004

- - goto

-

-

- 4

- - say

- 170000

-

-

- - return

-

-

- 50002

- - return

-

-

-

- - <

- 50000

- 150002

- 140001

- - gotoF

- 140001

-

- 19

- - return

-

-

- 50000

- - goto

-

-

- 137

- - '-'

- 50000

- 150001

- 100001

- - param

- 100001

- - 1

- 1

- 50000

- - SEND\_ATTR

- 0

-

- 0

- - SEND\_ATTR

- 1

-

- 1

- - SEND\_ATTR

- 2

-

- 2

- - SEND\_ATTR

- 3

-

- 3

- - SEND\_ATTR

- 4

-

- 4

- - SEND\_ATTR

- 5

-

- 5

- - SEND\_ATTR

- 6

-

- 6

- - SEND\_ATTR

- 7

-

- 7

- - SEND\_ATTR

- 8

-

- 8

- - SEND\_ATTR

- 9

-

- 9

- - SEND\_ATTR

- 10

-

- 10

- - SEND\_ATTR

- 11

-

- 11

- - SEND\_ATTR

- 12

-

- 12

- - SEND\_ATTR

- 13

-

- 13

- - SEND\_ATTR

- 14

-

- 14

- - SEND\_ATTR

- 15

-

- 15

- - SEND\_ATTR

- 16

-

- 16

- - SEND\_ATTR

- 17

-

- 17

- - SEND\_ATTR

- 18

-

- 18

- - SEND\_ATTR

- 19

-

- 19

- - SEND\_ATTR

- 20

-

- 20

- - SEND\_ATTR

- 21

-

- 21

- - SEND\_ATTR

- 22

-

- 22

- - SEND\_ATTR

- 23

-

- 23

- - SEND\_ATTR

- 24

-

- 24

- - SEND\_ATTR

- 25

-

- 25

- - SEND\_ATTR

- 26

-

- 26

- - SEND\_ATTR

- 27

-

- 27

- - SEND\_ATTR

- 28

-

- 28

- - SEND\_ATTR

- 29

-

- 29

- - SEND\_ATTR

- 30

-

- 30

- - SEND\_ATTR

- 31

-

- 31

- - SEND\_ATTR

- 32

-

- 32

- - SEND\_ATTR

- 33

-

- 33

- - SEND\_ATTR

- 34

-

- 34

- - SEND\_ATTR

- 35

-

- 35

- - SEND\_ATTR

- 36

-

- 36

- - SEND\_ATTR

- 37

-

- 37

- - SEND\_ATTR

- 38

-

- 38

- - SEND\_ATTR

- 39

-

- 39

- - SEND\_ATTR

- 40

-

- 40

- - SEND\_ATTR

- 41

-

- 41

- - SEND\_ATTR

- 42

-

- 42

- - SEND\_ATTR

- 43

-

- 43

- - SEND\_ATTR

- 44

-

- 44

- - SEND\_ATTR

- 45

-

- 45

- - SEND\_ATTR

- 46

-

- 46

- - SEND\_ATTR

- 47

-

- 47

- - SEND\_ATTR

- 48

-

- 48

- - SEND\_ATTR

- 49

-

- 49

- - SEND\_ATTR

- 50

-

- 50

- - SEND\_ATTR

- 51

-

- 51

- - SEND\_ATTR

- 52

-

- 52

- - SEND\_ATTR

- 53

-

- 53

- - SEND\_ATTR

- 54

-

- 54

- - gosub

- 15

-

- 50001

- - '-'

- 50000

- 150002

- 100002

- - param

- 100002

- - 1

- 1

- 50000

- - SEND\_ATTR

- 0

-

- 0

- - SEND\_ATTR

- 1

-

- 1

- - SEND\_ATTR

- 2

-

- 2

- - SEND\_ATTR

- 3

-

- 3

- - SEND\_ATTR

- 4

-

- 4

- - SEND\_ATTR

- 5

-

- 5

- - SEND\_ATTR

- 6

-

- 6

- - SEND\_ATTR

- 7

-

- 7

- - SEND\_ATTR

- 8

-

- 8

- - SEND\_ATTR

- 9

-

- 9

- - SEND\_ATTR

- 10

-

- 10

- - SEND\_ATTR

- 11

-

- 11

- - SEND\_ATTR

- 12

-

- 12

- - SEND\_ATTR

- 13

-

- 13

- - SEND\_ATTR

- 14

-

- 14

- - SEND\_ATTR

- 15

-

- 15

- - SEND\_ATTR

- 16

-

- 16

- - SEND\_ATTR

- 17

-

- 17

- - SEND\_ATTR

- 18

-

- 18

- - SEND\_ATTR

- 19

-

- 19

- - SEND\_ATTR

- 20

-

- 20

- - SEND\_ATTR

- 21

-

- 21

- - SEND\_ATTR

- 22

-

- 22

- - SEND\_ATTR

- 23

-

- 23

- - SEND\_ATTR

- 24

-

- 24

- - SEND\_ATTR

- 25

-

- 25

- - SEND\_ATTR

- 26

-

- 26

- - SEND\_ATTR

- 27

-

- 27

- - SEND\_ATTR

- 28

-

- 28

- - SEND\_ATTR

- 29

-

- 29

- - SEND\_ATTR

- 30

-

- 30

- - SEND\_ATTR

- 31

-

- 31

- - SEND\_ATTR

- 32

-

- 32

- - SEND\_ATTR

- 33

-

- 33

- - SEND\_ATTR

- 34

-

- 34

- - SEND\_ATTR

- 35

-

- 35

- - SEND\_ATTR

- 36

-

- 36

- - SEND\_ATTR

- 37

-

- 37

- - SEND\_ATTR

- 38

-

- 38

- - SEND\_ATTR

- 39

-

- 39

- - SEND\_ATTR

- 40

-

- 40

- - SEND\_ATTR

- 41

-

- 41

- - SEND\_ATTR

- 42

-

- 42

- - SEND\_ATTR

- 43

-

- 43

- - SEND\_ATTR

- 44

-

- 44

- - SEND\_ATTR

- 45

-

- 45

- - SEND\_ATTR

- 46

-

- 46

- - SEND\_ATTR

- 47

-

- 47

- - SEND\_ATTR

- 48

-

- 48

- - SEND\_ATTR

- 49

-

- 49

- - SEND\_ATTR

- 50

-

- 50

- - SEND\_ATTR

- 51

-

- 51

- - SEND\_ATTR

- 52

-

- 52

- - SEND\_ATTR

- 53

-

- 53

- - SEND\_ATTR

- 54

-

- 54

- - gosub

- 15

-

- 50002

- - +

- 50001

- 50002

- 100003

- - return

-

-

- 100003

- - return

-

-

-

- - say

- 170001

-

-

- - hear

- number

-

- 50000

- - param

- 50000

- - 1

- 1

- 50000

- - gosub

- 1

-

- 50001

- - param

- 50001

- - 1

- 1

- 50000

- - gosub

- 1

-

- 50002

- - say

- 50002

-

-

- - say

- 170002

-

-

- - param

- 50000

- - 1

- 1

- 50000

- - gosub

- 15

-

- 50003

- - say

- 50003

-

-

- - say

- 170003

-

-

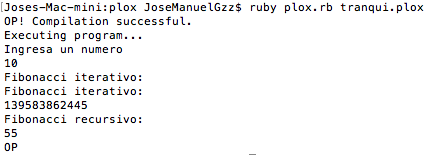
- - terminate

-

-

-

*Resultado del programa:*



#### **Prueba #8:** Programa que cuente con una división entre 0.

*Código:*

habemvs species EjemploObjectivePlox {

var open number a, b, c;

funk open oblivion chief() {

say("Introduce el valor de a");

hear(a);

say("Introduce el valor de b");

hear(b);

c = a / b;

say("El resultado de dividir a entre b es: ");

say(c);

}

}

*Resultado de código intermedio:*

---

- - goto

-

-

- 1

- - say

- 170000

-

-

- - hear

- number

-

- 0

- - say

- 170001

-

-

- - hear

- number

-

- 1

- - /

- 0

- 1

- 100000

- - =

-

- 100000

- 2

- - say

- 170002

-

-

- - say

- 2

-

-

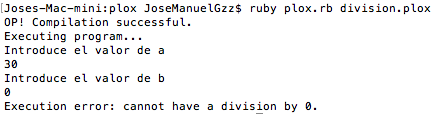
- - terminate

-

-

-

*Resultado del programa:*



#### **Prueba #7:** Programa con un error de sintaxis para que el compilador marque la línea del error y el tipo de error.

*Código:*

habemvs species EjemploObjectivePlox {

var open number a, b, c;

decimal d;

funk open oblivion chief() {

say("Introduce el valor de a");

hear(a);

say("Introduce el valor de b");

hear(b);

c = a / b;

say("El resultado de dividir a entre b es: ");

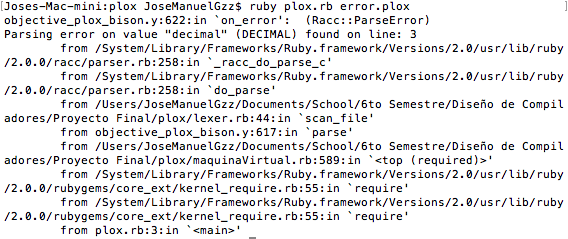
say(c);

}

}

*Resultado de código intermedio:* Dado que fue un error de sintaxis no se creó código intermedio

*Resultado del programa:*



#### **Prueba #10:** Un programa que ordene elementos dentro de un arreglo.

#### **Prueba #11:** Un programa con multiplicación de matrices.

*Código:*

habemvs species Arrgh {

var number arr[1..10], izq[1..3,1..3], der[1..3,1..3], res[1..3,1..3];

funk oblivion vamoAOrdenarlo(){

var number a,b, temp;

a = 1;

b = 1;

while(a<=10-1){

b = 1;

while(b<=10-a){

if(arr[b] > arr[b+1]){

temp = arr[b];

arr[b] = arr[b+1];

arr[b+1] = temp;

}

b += 1;

}

a += 1;

}

}

funk oblivion vamoAMultiplicarlo(){

var number a,b,c;

a = 1;

b = 1;

c = 1;

while(a<=3){

while(b<=3){

while(c<=3){

res[a,c] += (der[a,c]\*izq[c,b]);

c += 1;

}

c = 1;

b +=1;

}

b = 1;

a += 1;

}

}

funk oblivion chief(){

var number a,b, temp;

a = 1;

b = 1;

while(a<=10){

say("Ingresa el siguiente numero del arreglo a ordenar");

hear(arr[a]);

a+= 1;

}

vamoAOrdenarlo();

say("Arreglo ordenado:");

say(arr[1]);

say(arr[2]);

say(arr[3]);

say(arr[4]);

say(arr[5]);

say(arr[6]);

say(arr[7]);

say(arr[8]);

say(arr[9]);

say(arr[10]);

a = 1;

b = 1;

while(a<=3){

b = 1;

while(b<=3){

say("Ingresa el siguiente numero de la primera matriz a multiplicar");

hear(izq[a,b]);

b += 1;

}

a += 1;

}

a = 1;

b = 1;

while(a<=3){

b = 1;

while(b<=3){

say("Ingresa el siguiente numero de la segunda matriz a multiplicar");

hear(der[a,b]);

b += 1;

}

a += 1;

}

a = 1;

b = 1;

while(a<=3){

while(b<=3){

res[a,b] = 0;

b += 1;

}

b = 1;

a += 1;

}

vamoAMultiplicarlo();

a = 1;

b = 1;

say("Matriz resultado");

while(a<=3){

while(b<=3){

say(res[a,b]);

b += 1;

}

b = 1;

a += 1;

}

}

}

*Resultado de código intermedio:*

---

- - goto

-

-

- 79

- - =

-

- 150000

- 50000

- - =

-

- 150000

- 50001

- - '-'

- 150001

- 150000

- 100000

- - <=

- 50000

- 100000

- 140000

- - gotoF

- 140000

-

- 40

- - =

-

- 150000

- 50001

- - '-'

- 150001

- 50000

- 100001

- - <=

- 50001

- 100001

- 140001

- - gotoF

- 140001

-

- 38

- - verify

- 10

- 1

- 50001

- - +SpecialRight

- 50001

- -1

- 100002

- - +SpecialRight

- 100002

- 0

- 100003

- - +

- 50001

- 150000

- 100004

- - verify

- 10

- 1

- 100004

- - +SpecialRight

- 100004

- -1

- 100005

- - +SpecialRight

- 100005

- 0

- 100006

- - '>'

- 100003.0

- 100006.0

- 140002

- - gotoF

- 140002

-

- 36

- - verify

- 10

- 1

- 50001

- - +SpecialRight

- 50001

- -1

- 100007

- - +SpecialRight

- 100007

- 0

- 100008

- - =

-

- 100008.0

- 50002

- - verify

- 10

- 1

- 50001

- - +SpecialRight

- 50001

- -1

- 100009

- - +SpecialRight

- 100009

- 0

- 100010

- - +

- 50001

- 150000

- 100011

- - verify

- 10

- 1

- 100011

- - +SpecialRight

- 100011

- -1

- 100012

- - +SpecialRight

- 100012

- 0

- 100013

- - =

-

- 100013.0

- 100010.0

- - +

- 50001

- 150000

- 100014

- - verify

- 10

- 1

- 100014

- - +SpecialRight

- 100014

- -1

- 100015

- - +SpecialRight

- 100015

- 0

- 100016

- - =

-

- 50002

- 100016.0

- - +=

-

- 150000

- 50001

- - goto

-

-

- 7

- - +=

-

- 150000

- 50000

- - goto

-

-

- 3

- - return

-

-

-

- - =

-

- 150000

- 50000

- - =

-

- 150000

- 50001

- - =

-

- 150000

- 50002

- - <=

- 50000

- 150002

- 140003

- - gotoF

- 140003

-

- 78

- - <=

- 50001

- 150002

- 140004

- - gotoF

- 140004

-

- 75

- - <=

- 50002

- 150002

- 140005

- - gotoF

- 140005

-

- 72

- - verify

- 3

- 1

- 50000

- - '\*SpecialRight'

- 50000

- 3

- 100017

- - verify

- 3

- 1

- 50002

- - +

- 50002

- 100017

- 100018

- - +SpecialRight

- 100018

- -4

- 100019

- - +SpecialRight

- 100019

- 28

- 100020

- - verify

- 3

- 1

- 50000

- - '\*SpecialRight'

- 50000

- 3

- 100021

- - verify

- 3

- 1

- 50002

- - +

- 50002

- 100021

- 100022

- - +SpecialRight

- 100022

- -4

- 100023

- - +SpecialRight

- 100023

- 19

- 100024

- - verify

- 3

- 1

- 50002

- - '\*SpecialRight'

- 50002

- 3

- 100025

- - verify

- 3

- 1

- 50001

- - +

- 50001

- 100025

- 100026

- - +SpecialRight

- 100026

- -4

- 100027

- - +SpecialRight

- 100027

- 10

- 100028

- - '\*'

- 100024.0

- 100028.0

- 100029

- - +=

-

- 100029

- 100020.0

- - +=

-

- 150000

- 50002

- - goto

-

-

- 48

- - =

-

- 150000

- 50002

- - +=

-

- 150000

- 50001

- - goto

-

-

- 46

- - =

-

- 150000

- 50001

- - +=

-

- 150000

- 50000

- - goto

-

-

- 44

- - return

-

-

-

- - =

-

- 150000

- 50000

- - =

-

- 150000

- 50001

- - <=

- 50000

- 150001

- 140006

- - gotoF

- 140006

-

- 90

- - say

- 170000

-

-

- - verify

- 10

- 1

- 50000

- - +SpecialRight

- 50000

- -1

- 100030

- - +SpecialRight

- 100030

- 0

- 100031

- - hear

- number

-

- 100031.0

- - +=

-

- 150000

- 50000

- - goto

-

-

- 81

- - SEND\_ATTR

- 0

-

- 0

- - SEND\_ATTR

- 1

-

- 1

- - SEND\_ATTR

- 2

-

- 2

- - SEND\_ATTR

- 3

-

- 3

- - SEND\_ATTR

- 4

-

- 4

- - SEND\_ATTR

- 5

-

- 5

- - SEND\_ATTR

- 6

-

- 6

- - SEND\_ATTR

- 7

-

- 7

- - SEND\_ATTR

- 8

-

- 8

- - SEND\_ATTR

- 9

-

- 9

- - SEND\_ATTR

- 10

-

- 10

- - SEND\_ATTR

- 11

-

- 11

- - SEND\_ATTR

- 12

-

- 12

- - SEND\_ATTR

- 13

-

- 13

- - SEND\_ATTR

- 14

-

- 14

- - SEND\_ATTR

- 15

-

- 15

- - SEND\_ATTR

- 16

-

- 16

- - SEND\_ATTR

- 17

-

- 17

- - SEND\_ATTR

- 18

-

- 18

- - SEND\_ATTR

- 19

-

- 19

- - SEND\_ATTR

- 20

-

- 20

- - SEND\_ATTR

- 21

-

- 21

- - SEND\_ATTR

- 22

-

- 22

- - SEND\_ATTR

- 23

-

- 23

- - SEND\_ATTR

- 24

-

- 24

- - SEND\_ATTR

- 25

-

- 25

- - SEND\_ATTR

- 26

-

- 26

- - SEND\_ATTR

- 27

-

- 27

- - SEND\_ATTR

- 28

-

- 28

- - SEND\_ATTR

- 29

-

- 29

- - SEND\_ATTR

- 30

-

- 30

- - SEND\_ATTR

- 31

-

- 31

- - SEND\_ATTR

- 32

-

- 32

- - SEND\_ATTR

- 33

-

- 33

- - SEND\_ATTR

- 34

-

- 34

- - SEND\_ATTR

- 35

-

- 35

- - SEND\_ATTR

- 36

-

- 36

- - gosub

- 1

-

-

- - say

- 170001

-

-

- - verify

- 10

- 1

- 150000

- - +SpecialRight

- 150000

- -1

- 100032

- - +SpecialRight

- 100032

- 0

- 100033

- - say

- 100033.0

-

-

- - verify

- 10

- 1

- 150003

- - +SpecialRight

- 150003

- -1

- 100034

- - +SpecialRight

- 100034

- 0

- 100035

- - say

- 100035.0

-

-

- - verify

- 10

- 1

- 150002

- - +SpecialRight

- 150002

- -1

- 100036

- - +SpecialRight

- 100036

- 0

- 100037

- - say

- 100037.0

-

-

- - verify

- 10

- 1

- 150004

- - +SpecialRight

- 150004

- -1

- 100038

- - +SpecialRight

- 100038

- 0

- 100039

- - say

- 100039.0

-

-

- - verify

- 10

- 1

- 150005

- - +SpecialRight

- 150005

- -1

- 100040

- - +SpecialRight

- 100040

- 0

- 100041

- - say

- 100041.0

-

-

- - verify

- 10

- 1

- 150006

- - +SpecialRight

- 150006

- -1

- 100042

- - +SpecialRight

- 100042

- 0

- 100043

- - say

- 100043.0

-

-

- - verify

- 10

- 1

- 150007

- - +SpecialRight

- 150007

- -1

- 100044

- - +SpecialRight

- 100044

- 0

- 100045

- - say

- 100045.0

-

-

- - verify

- 10

- 1

- 150008

- - +SpecialRight

- 150008

- -1

- 100046

- - +SpecialRight

- 100046

- 0

- 100047

- - say

- 100047.0

-

-

- - verify

- 10

- 1

- 150009

- - +SpecialRight

- 150009

- -1

- 100048

- - +SpecialRight

- 100048

- 0

- 100049

- - say

- 100049.0

-

-

- - verify

- 10

- 1

- 150001

- - +SpecialRight

- 150001

- -1

- 100050

- - +SpecialRight

- 100050

- 0

- 100051

- - say

- 100051.0

-

-

- - =

-

- 150000

- 50000

- - =

-

- 150000

- 50001

- - <=

- 50000

- 150002

- 140007

- - gotoF

- 140007

-

- 188

- - =

-

- 150000

- 50001

- - <=

- 50001

- 150002

- 140008

- - gotoF

- 140008

-

- 186

- - say

- 170002

-

-

- - verify

- 3

- 1

- 50000

- - '\*SpecialRight'

- 50000

- 3

- 100052

- - verify

- 3

- 1

- 50001

- - +

- 50001

- 100052

- 100053

- - +SpecialRight

- 100053

- -4

- 100054

- - +SpecialRight

- 100054

- 10

- 100055

- - hear

- number

-

- 100055.0

- - +=

-

- 150000

- 50001

- - goto

-

-

- 174

- - +=

-

- 150000

- 50000

- - goto

-

-

- 171

- - =

-

- 150000

- 50000

- - =

-

- 150000

- 50001

- - <=

- 50000

- 150002

- 140009

- - gotoF

- 140009

-

- 207

- - =

-

- 150000

- 50001

- - <=

- 50001

- 150002

- 140010

- - gotoF

- 140010

-

- 205

- - say

- 170003

-

-

- - verify

- 3

- 1

- 50000

- - '\*SpecialRight'

- 50000

- 3

- 100056

- - verify

- 3

- 1

- 50001

- - +

- 50001

- 100056

- 100057

- - +SpecialRight

- 100057

- -4

- 100058

- - +SpecialRight

- 100058

- 19

- 100059

- - hear

- number

-

- 100059.0

- - +=

-

- 150000

- 50001

- - goto

-

-

- 193

- - +=

-

- 150000

- 50000

- - goto

-

-

- 190

- - =

-

- 150000

- 50000

- - =

-

- 150000

- 50001

- - <=

- 50000

- 150002

- 140011

- - gotoF

- 140011

-

- 225

- - <=

- 50001

- 150002

- 140012

- - gotoF

- 140012

-

- 222

- - verify

- 3

- 1

- 50000

- - '\*SpecialRight'

- 50000

- 3

- 100060

- - verify

- 3

- 1

- 50001

- - +

- 50001

- 100060

- 100061

- - +SpecialRight

- 100061

- -4

- 100062

- - +SpecialRight

- 100062

- 28

- 100063

- - =

-

- 150010

- 100063.0

- - +=

-

- 150000

- 50001

- - goto

-

-

- 211

- - =

-

- 150000

- 50001

- - +=

-

- 150000

- 50000

- - goto

-

-

- 209

- - SEND\_ATTR

- 0

-

- 0

- - SEND\_ATTR

- 1

-

- 1

- - SEND\_ATTR

- 2

-

- 2

- - SEND\_ATTR

- 3

-

- 3

- - SEND\_ATTR

- 4

-

- 4

- - SEND\_ATTR

- 5

-

- 5

- - SEND\_ATTR

- 6

-

- 6

- - SEND\_ATTR

- 7

-

- 7

- - SEND\_ATTR

- 8

-

- 8

- - SEND\_ATTR

- 9

-

- 9

- - SEND\_ATTR

- 10

-

- 10

- - SEND\_ATTR

- 11

-

- 11

- - SEND\_ATTR

- 12

-

- 12

- - SEND\_ATTR

- 13

-

- 13

- - SEND\_ATTR

- 14

-

- 14

- - SEND\_ATTR

- 15

-

- 15

- - SEND\_ATTR

- 16

-

- 16

- - SEND\_ATTR

- 17

-

- 17

- - SEND\_ATTR

- 18

-

- 18

- - SEND\_ATTR

- 19

-

- 19

- - SEND\_ATTR

- 20

-

- 20

- - SEND\_ATTR

- 21

-

- 21

- - SEND\_ATTR

- 22

-

- 22

- - SEND\_ATTR

- 23

-

- 23

- - SEND\_ATTR

- 24

-

- 24

- - SEND\_ATTR

- 25

-

- 25

- - SEND\_ATTR

- 26

-

- 26

- - SEND\_ATTR

- 27

-

- 27

- - SEND\_ATTR

- 28

-

- 28

- - SEND\_ATTR

- 29

-

- 29

- - SEND\_ATTR

- 30

-

- 30

- - SEND\_ATTR

- 31

-

- 31

- - SEND\_ATTR

- 32

-

- 32

- - SEND\_ATTR

- 33

-

- 33

- - SEND\_ATTR

- 34

-

- 34

- - SEND\_ATTR

- 35

-

- 35

- - SEND\_ATTR

- 36

-

- 36

- - gosub

- 41

-

-

- - =

-

- 150000

- 50000

- - =

-

- 150000

- 50001

- - say

- 170004

-

-

- - <=

- 50000

- 150002

- 140013

- - gotoF

- 140013

-

- 282

- - <=

- 50001

- 150002

- 140014

- - gotoF

- 140014

-

- 279

- - verify

- 3

- 1

- 50000

- - '\*SpecialRight'

- 50000

- 3

- 100064

- - verify

- 3

- 1

- 50001

- - +

- 50001

- 100064

- 100065

- - +SpecialRight

- 100065

- -4

- 100066

- - +SpecialRight

- 100066

- 28

- 100067

- - say

- 100067.0

-

-

- - +=

-

- 150000

- 50001

- - goto

-

-

- 268

- - =

-

- 150000

- 50001

- - +=

-

- 150000

- 50000

- - goto

-

-

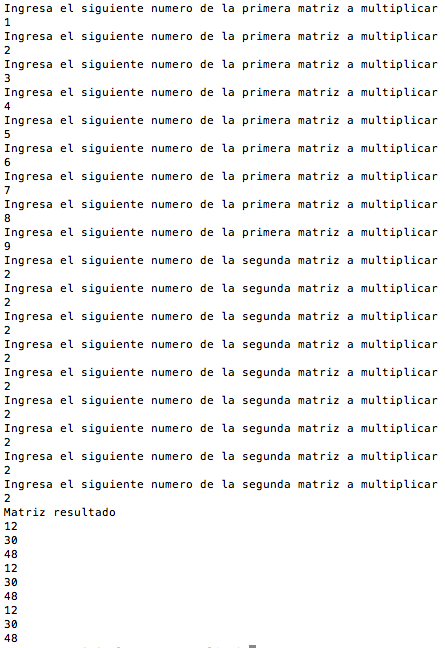
- 266

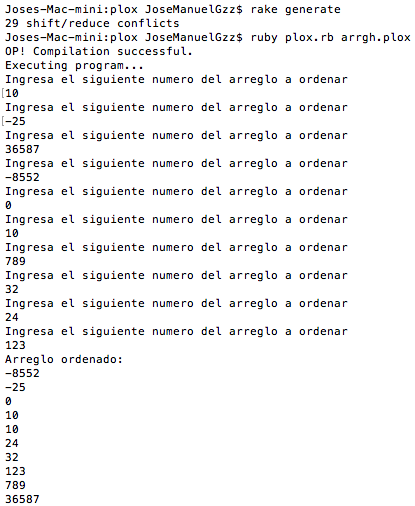
- - terminate

-

-

-

*Resultado del programa:*



#### **Prueba #12:** Un programa con un error de semántica.

*Código:*

habemvs species EjemploObjectivePlox {

var open number a, b, c;

funk oblivion fibonacci() {

}

funk number fibonacci() {

}

funk open oblivion chief() {

say("Introduce el valor de a");

hear(a);

say("Introduce el valor de b");

hear(b);

c = a / b;

say("El resultado de dividir a entre b es: ");

say(c);

}

}

*Resultado de código intermedio:* Dado que hay un error semántico no se generan cuádruplos ya que se limpian todas las estructuras.

*Resultado del programa:*



# F) Listados documentados del proyecto

En esta parte se muestran algunos de los métodos más importantes con sus respectivos comentarios, indicando qué hacen y qué parámetros reciben si es que reciben alguno. Estos se van poniendo en el orden en el que fueron creados, es decir, primero lo que se hizo en fase de compilación y luego lo que se hizo en la máquina virtual. Cabe mencionar que para la máquina virtual solo se van a poner algunos de los métodos ya que muchos tienen la misma estructura.

## Directorio de procedimientos

# Método que crea una nueva clase en el hash correspondiente.

  # Parámetros de entrada:

  # species: el nombre de la clase

  # Este método crea la estructura del hash y lo agrega

  # al dirProc. Se muestra un error si la clase ya existe.

def newSpecies(species)

if $speciesBook[species] == nil

$speciesBook[species] = Hash.new

$speciesBook[species]["methods"] = Hash.new

$speciesBook[species]["variables"] = Hash.new

$speciesBook[species]["size"] = { "number" => 0, "decimal" => 0, "string" => 0, "char" => 0, "logic" => 0 }

$actualSpecies = species

$actualMethod = "species"

resetCounters("global")

else

abort("Semantic error: species '#{species}' is already defined. Error on line: #{$line\_number}")

end

end

# Método que implementa la herencia entre clases.

  # Parámetros de entrada:

  # father: El nombre de la clase padre

  # Este método agrega el padre de una clase a la que se encuentra

  # se hereda y muestra un mensaje de error si la clase padre

  # no se ha creado.

def heirSpecies(father)

if $speciesBook[father] != nil

$speciesBook[$actualSpecies]["father"] = $speciesBook[father]

$magicCounter["global"] = $speciesBook[father]["size"].clone

else

abort("Semantic error: '#{father}' father of species '#{$actualSpecies}' is not defined. Error on line: #{$line\_number}")

end

end

# Método que añade una variable a la tabla de variables para una determinada

  # clase.

  # Parámetros de entrada:

  # Id: El nombre de la variable

  # Este método crea una nueva variable y se añade

  # a la tabla de variables. Tiene múltiples validaciones

  # en caso de que la clase de la variable que estas creando no existe,

  # o un error si no se define la variable,

  # O si se ha definido previamente.

def newVariable(id)

if $actualMethod == "species"

unless idDeclaredInSpeciesRecursively($speciesBook[$actualSpecies], id, "variables")

abort("Semantic error: species '#{$actualType}' is not defined. Error on line: #{$line\_number}") unless isValidType($actualType)

abort("Semantic error: you cannot have recursive species definitions. Error on line: #{$line\_number}") if $actualType == $actualSpecies

$speciesBook[$actualSpecies]["variables"][id] = Hash.new

$speciesBook[$actualSpecies]["variables"][id]["type"] = $actualType

$speciesBook[$actualSpecies]["variables"][id]["scope"] = $actualModifier

$speciesBook[$actualSpecies]["variables"][id]["modifiable"] = $isVariable

if $actualType == "number" || $actualType == "decimal" || $actualType == "string" || $actualType == "char" || $actualType == "logic"

$speciesBook[$actualSpecies]["variables"][id]["location"] = locationGenerator(1, "global", $actualType)

$speciesBook[$actualSpecies]["size"][$actualType] += 1

else

createAtributtesRecursively(id, $speciesBook[$actualType])

end

else

abort("Semantic error: variable '#{id}' is already defined. Error on line: #{$line\_number}")

end

else

if $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id] == nil

abort("Semantic error: species '#{$actualType}' is not defined. Error on line: #{$line\_number}") unless isValidType($actualType)

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id] = Hash.new

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id]["type"] = $actualType

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id]["modifiable"] = $isVariable

if $actualType == "number" || $actualType == "decimal" || $actualType == "string" || $actualType == "char" || $actualType == "logic"

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id]["location"] = locationGenerator(1, "local", $actualType)

$speciesBook[$actualSpecies]["methods"][$actualMethod]["size"][$actualType] += 1

end

else

abort("Semantic error: variable '#{id}' is already defined in method '#{$actualMethod}'. Error on line: #{$line\_number}")

end

end

$actualVarId = id

end

## Arreglos

# Método que crea una nueva matriz. Se recupera la species y el

  # tipo de la matriz. Se lleva a cabo una validación para ver si está

  # tratando de crear una matriz de objetos (que no es compatible).

def newArray()

if $actualMethod == "species"

if $actualType == "number" || $actualType == "decimal" || $actualType == "string" || $actualType == "char" || $actualType == "logic"

$speciesBook[$actualSpecies]["variables"][$actualVarId]["dimensions"] = Array.new

else

abort("Semantic error: ObjectivePlox currently doesnt support arrays of objects. Error on line: #{$line\_number}")

end

else

if $actualType == "number" || $actualType == "decimal" || $actualType == "string" || $actualType == "char" || $actualType == "logic"

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][$actualVarId]["dimensions"] = Array.new

else

abort("Semantic error: ObjectivePlox currently doesnt support arrays of objects. Error on line: #{$line\_number}")

end

end

end

# Método que agrega una nueva dimension en la declaracion de una matriz.

# Parámetros de entrada:

# infLi: Limite inferior

# supLi: Limite superior

# Crea un hash con los datos de los limites y la r acumulada hasta el momento

# y lo agrega a las dimensiones de la variable

def newDimension(infLi, supLi)

dimension = Hash.new

dimension["sl"] = supLi[1]

dimension["il"] = infLi[1]

dimension["r"] = supLi[1] - infLi[1] + 1

if $actualMethod == "species"

dimension["r"] \*= $speciesBook[$actualSpecies]["variables"][$actualVarId]["dimensions"].last["r"] if $speciesBook[$actualSpecies]["variables"][$actualVarId]["dimensions"].last != nil

$speciesBook[$actualSpecies]["variables"][$actualVarId]["dimensions"].push(dimension)

else

dimension["r"] \*= $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][$actualVarId]["dimensions"].last["r"] if $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][$actualVarId]["dimensions"].last != nil

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][$actualVarId]["dimensions"].push(dimension)

end

end

# Método que saca la dimension total de un array y la k

# Segun la formula vuelve a recorrer toda la lista

# de dimesiones para sacar el tamaño total y la k

def defineArray()

suma = 0

if $actualMethod == "species"

r = $speciesBook[$actualSpecies]["variables"][$actualVarId]["dimensions"].last["r"]

$speciesBook[$actualSpecies]["variables"][$actualVarId]["totalsize"] = r

locationGenerator(r-1, "global", $actualType)

$speciesBook[$actualSpecies]["variables"][$actualVarId]["dimensions"].each\_with\_index do |h, i|

m = r / (h["sl"] - h["il"] + 1)

r = m

suma += (h["il"] \* m)

if i == $speciesBook[$actualSpecies]["variables"][$actualVarId]["dimensions"].count - 1

$speciesBook[$actualSpecies]["variables"][$actualVarId]["dimensions"][i]["k"] = - suma

else

$speciesBook[$actualSpecies]["variables"][$actualVarId]["dimensions"][i]["m"] = m

end

end

else

r = $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][$actualVarId]["dimensions"].last["r"]

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][$actualVarId]["totalsize"] = r

locationGenerator(r-1, "local", $actualType)

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][$actualVarId]["dimensions"].each\_with\_index do |h, i|

m = r / (h["sl"] - h["il"] + 1)

r = m

suma += (h["il"] \* m)

if i == $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][$actualVarId]["dimensions"].count - 1

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][$actualVarId]["dimensions"][i]["k"] = - suma

else

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][$actualVarId]["dimensions"][i]["m"] = m

end

end

end

end

# Método que valida que genera los cuadruplos necesarios

# para validar el rango de una llamada a un arreglo

# y las operaciones necesarias del indexamiento

def newDimensionIndex(index)

abort("Semantic error: you can only use numbers to refer a dimension index. Error on line: #{$line\_number}") unless index[0] == "number"

$arrayIndexStack.last["index"] += 1

abort("Semantic error: wrong nunmber of dimensions, the variable you're accessing has less dimensions. Error on line: #{$line\_number}") unless $arrayIndexStack.last["index"] < $arrayIndexStack.last["dimensions"].count

sl = $arrayIndexStack.last["dimensions"][$arrayIndexStack.last["index"]]["sl"]

il = $arrayIndexStack.last["dimensions"][$arrayIndexStack.last["index"]]["il"]

$quadrupleVector.push(["verify", sl, il, index[1]])

temp = -1

if $arrayIndexStack.last["index"] != $arrayIndexStack.last["dimensions"].count - 1

temp = locationGenerator(1,"temporal","number")

$quadrupleVector.push(["\*SpecialRight", index[1], $arrayIndexStack.last["dimensions"][$arrayIndexStack.last["index"]]["m"], temp])

if $arrayIndexStack.last["index"] > 0

aux = temp

temp = locationGenerator(1,"temporal","number")

$quadrupleVector.push(["+", aux, $arrayIndexStack.last["temporal"], temp])

$arrayIndexStack.last["temporal"] = temp

end

elsif $arrayIndexStack.last["index"] > 0

temp = locationGenerator(1,"temporal","number")

$quadrupleVector.push(["+", index[1], $arrayIndexStack.last["temporal"], temp])

$arrayIndexStack.last["temporal"] = temp

else

temp = index[1]

end

$arrayIndexStack.last["temporal"] = temp

end

## Validaciones semánticas de herencia

# Método recursivo que regresa si un id está declarado en una

# species o en alguno de los padres.

# Parámetros de entrada:

# species: hash de la species a buscar

# id: un string del id que se está buscando

# type: el tipo del id que se está buscando

def idDeclaredInSpeciesRecursively(species, id, type)

if species[type][id] != nil # regresa si la variable ya existe

return true

elsif species["father"] == nil # si la clase no tiene padre

return false

else

return idDeclaredInSpeciesRecursively(species["father"], id, type) # checa para su padre

end

end

## Funciones

# Método para dar de alta una funcion o método

# en una species.

# Parámetros de entrada:

# id: nombre de la funcion

# Este método crea una nueva funcion y se añade

# a la tabla de variables. Tiene múltiples validaciones

# en caso de que la clase de la funcion que estas creando no existe,

# o un error si no se define el tipo de la funcion,

# O si la funcion ya se ha definido previamente.

def newMethod(id)

unless $speciesBook[$actualSpecies]["methods"][id] != nil

unless isValidType($actualType) || $actualType == "oblivion"

abort("Semantic error: species '#{$actualType}' is not defined. Error on line: #{$line\_number}")

end

$speciesBook[$actualSpecies]["methods"][id] = Hash.new

$speciesBook[$actualSpecies]["methods"][id]["type"] = $actualType

$speciesBook[$actualSpecies]["methods"][id]["scope"] = $actualModifier

$speciesBook[$actualSpecies]["methods"][id]["size"] = { "number" => 0, "decimal" => 0, "string" => 0, "char" => 0, "logic" => 0 }

$speciesBook[$actualSpecies]["methods"][id]["variables"] = Hash.new

$speciesBook[$actualSpecies]["methods"][id]["argumentList"] = []

$speciesBook[$actualSpecies]["methods"][id]["begin"] = $quadrupleVector.count()

$actualMethod = id

resetCounters("local")

else

abort("Semantic error: variable '#{id}' is already defined. Error on line: #{$line\_number}")

end

end

# Método que agrega una nuevo argumento a la funcion actual.

# Parámetros de entrada:

# type: tipo de la funcion

# id: el id del argumento

# Da de alta el argumento en la tabla de variables de la funcion

# e incrementa el contador de parametros

def newArgument(type, id)

if $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id] == nil

unless isValidType(type)

abort("Semantic error: species '#{type}' is not defined. Error on line: #{$line\_number}")

end

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id] = Hash.new

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id]["type"] = $actualType

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id]["modifiable"] = $isVariable

if $actualType == "number" || $actualType == "decimal" || $actualType == "string" || $actualType == "char" || $actualType == "logic"

$speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id]["location"] = locationGenerator(1, "local", $actualType)

$speciesBook[$actualSpecies]["methods"][$actualMethod]["size"][$actualType] += 1

end

arg = { "location" => $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id]["location"], "type" => type }

$speciesBook[$actualSpecies]["methods"][$actualMethod]["argumentList"].push(arg)

else

abort("Semantic error: argument '#{id}' is already defined in method '#{$actualMethod}'. Error on line: #{$line\_number}")

end

end

# Método que se encarga de mandar todos los atributos

# de la clase correspondiente al iniciar la llamada a una funcion

def sendAttributes()

$speciesBook[$actualSpecies]["variables"].each do |key, h|

tokens = key.split(".")

type = h["type"]

isPrimitive = type == "number" || type == "decimal" || type == "char" || type == "string" || type == "logic"

if tokens[0] == $actualId && tokens.count > 1 && isPrimitive

destination = idLocationRecursively($speciesBook[$speciesBook[$actualSpecies]["variables"][tokens[0]]["type"]], key[key.index(".")+1..-1])

quadruple = ["SEND\_ATTR", destination, nil, h["location"]]

$quadrupleVector.push(quadruple)

if h["totalsize"] != nil

for index in 1..h["totalsize"]-1

quadruple = ["SEND\_ATTR", destination, nil, h["location"]+index]

$quadrupleVector.push(quadruple)

end

end

end

end

end

# Método que se encarga de validar que una funcion haya sido

# llamada correctamente

def validateFunk()

if $argumentCount != 0

$argumentCountStack.push($argumentCount)

end

$argumentCount = 0

$funkGlobalContext = false

if $actualIdSpecies == nil

$actualIdSpecies = $actualSpecies

$funkGlobalContext = true

end

$speciesStack.push($actualIdSpecies)

$idStack.push($actualIdFunk)

funkHash = speciesHashOfFunkRecursively($speciesBook[$actualIdSpecies], $actualIdFunk)

if funkHash != nil

if funkHash["scope"] || $funkGlobalContext

return funkHash["type"]

else

abort("Semantic error: method #{$actualIdFunk} is not open. Error on line: #{$line\_number}")

end

else

abort("Semantic error: method #{$actualIdFunk} not defined. Error on line: #{$line\_number}")

end

end

## Constantes, direcciones de memoria y validación del cubo semántico

# Método que checa los tipos de 2 operandos

# con el cubo semantico.

def expressionResultType(operator, leftOp, rightOp)

if $semanticCube[leftOp][rightOp][operator] == nil

abort("Semantic error: type mismatch. Cannot combine type '#{leftOp}' and type '#{rightOp}' with operator '#{operator}'. Error on line: #{$line\_number}")

end

return $semanticCube[leftOp][rightOp][operator]

end

# Método que da de alta una constante en la

# tabla de constantes

def newConstant(type, value)

if $constantBook[value] == nil

$constantBook[value] = locationGenerator(1, "constant", type)

end

return $constantBook[value]

end

# Método que genera direcciones de memoria segun

# el scope, el tipo y el tamaño de la variable

def locationGenerator(size, scope, type)

if $theMagicNumber - $magicCounter[scope][type] >= size

location = $magicReference[scope][type] \* $theMagicNumber + $magicCounter[scope][type]

$magicCounter[scope][type] += size

return location

else

abort("Compilation error: out of memory for a #{type} variable with the #{scope} scope. While compiling line: #{$line\_number}")

end

end

## Recuperación de valores

# Metódo que regresa la ubicación de una variable en memoria

# segun su id

def retrieveIdLocation(id)

if $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id] != nil

return $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id]["location"]

elsif idLocationRecursively($speciesBook[$actualSpecies], id) != nil

return idLocationRecursively($speciesBook[$actualSpecies], id)

else

abort("Semantic error: variable '#{id}' not declared. Error on line: #{$line\_number}")

end

end

# Metódo que regresa el tipo de una variable

# segun su id

def retrieveIdType(id)

if $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id] != nil

return $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id]["type"]

elsif idTypeRecursively($speciesBook[$actualSpecies], id) != nil

return idTypeRecursively($speciesBook[$actualSpecies], id)

else

abort("Semantic error: variable '#{id}' not declared. Error on line: #{$line\_number}")

end

end

# Metódo que regresa el hash con todas las dimensiones

# de una varible segun su id

def retrieveIdDimensions(id)

if $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id] != nil

return $speciesBook[$actualSpecies]["methods"][$actualMethod]["variables"][id]["dimensions"]

elsif idDimensionsRecursively($speciesBook[$actualSpecies], id) != nil

return idDimensionsRecursively($speciesBook[$actualSpecies], id)

else

abort("Semantic error: dimensioned variable '#{id}' not declared. Error on line: #{$line\_number}")

end

end

## Máquina virtual

# Class that simulates the memory call stack and stores

# parameters and attributes and has a context identifier

class Memory

attr\_accessor :params, :attributes, :context

def initialize()

@params = Hash.new

@attributes = Hash.new

@params["paramsCompleted"] = false

@context = -1

end

end

# Class that performs the necesary operations

# for execution. AKA Virtual Machine

class VirtualMachine

# The following methods, initialize and initialContext are

# the ones that simulate the actual memory.

# Method that initializes all the variables for the

# virtual machine class and especially the memory where

# directions and values are going to be stored.

def initialize()

# Variable that saves the number of the current Quadruple

@quadruplePointer = 0

@constantBook = YAML.load\_file('constants.yaml')

@quadrupleVector = YAML.load\_file('quadruples.yaml')

@memory = Array.new

@contextPointer = -1

@callStack = Array.new

@callPointer = -1

initialContext()

end

# Method that defines the initial context.

# It pushes the context and increases the context

# pointer size in case a new context arrives later.

# This method is called after initialize() and it helps

# keep track of the memory used during execution.

def initialContext()

context = Hash.new

@constantBook.each do |key, value|

context[key] = value

end

context["isSwapping"] = false

@memory.push(context)

@contextPointer += 1

end

# Method that creates a new context.

# It initializes all variables and

# it stores the context, it retrieves

# the returning direction, the returning quadruple

# and it pushes the context to the memory.

def newContext()

context = Hash.new

@constantBook.each do |key, value|

context[key] = value

end

context["isSwapping"] = false

call = @callStack.pop()

@callPointer -= 1

call.params.each do |key, value|

context[key] = value

end

call.attributes.each do |key, h|

context[key] = h["value"]

end

context["forReturn"] = Marshal.load(Marshal.dump(call.attributes))

context["returnDir"] = @quadrupleVector[@quadruplePointer][3]

context["returningQuadruple"] = @quadruplePointer

@memory.push(context)

@contextPointer += 1

@quadruplePointer = @quadrupleVector[@quadruplePointer][1] - 1

end

# Method that performs a change of context

# and pushes the last context to the call

# stack.

def newCall()

call = Memory.new

call.context = $contextPointer

@callStack.push(call)

@callPointer += 1

end

# Method that returns the value of a variable

# given its address in memory.

# Entry parameters:

# dir - The direction to where the variable and value

# are stored in memory

def obtainData(dir)

if (dir.is\_a?(Integer))

return @memory[@contextPointer][dir]

else

return @memory[@contextPointer][@memory[@contextPointer][dir.to\_i]]

end

end

# Method that stores a direction and value in memory.

# Entry parameters:

# dir - The direction to be stored

# value - The value to be stored for the direction

def store(dir, value)

if (dir.is\_a?(Integer))

@memory[@contextPointer][dir] = value

else

@memory[@contextPointer][@memory[@contextPointer][dir.to\_i]] = value

end

end

# Method that corresponds to the return action.

# it gets the direction to which the code needs to

# return after being in a function.

def returnAction

past = @memory.pop()

@contextPointer -= 1

past["forReturn"].each do |key, h|

@memory[@contextPointer][h["return"]] = past[key]

end

@memory[@contextPointer][past["returnDir"]] = past[@quadrupleVector[@quadruplePointer][3]]

@quadruplePointer = past["returningQuadruple"]

if @callStack.count == 0

@memory[@contextPointer]["isSwapping"] = false

elsif @callStack[@callPointer].context != @contextPointer

@memory[@contextPointer]["isSwapping"] = false

end

end

# Method that executes quadruples with a switch.

# Entry parameters: The pointer to the current quadruple.

# It obtains the first element of the current quadruple (it's action)

# and it executes the method that corresponds to that action.

def executeQuadruple(quadruple)

case quadruple[0]

when "+"

addition

when "-"

subtraction

when "\*"

multiplication

when "/"

division

when "%"

mod

when ">"

greaterThan

when "<"

lessThan

when ">="

greaterEqualThan

when "<="

lessEqualThan

when "=="

equality

when "!="

different

when "!"

notExp

when "&&"

andExp

when "||"

orExp

when "="

equal

when "+="

additionAssign

when "-="

subtractionAssign

when "\*="

multiplicationAssign

when "/="

divisionAssign

when "||="

orAssign

when "&&="

andAssign

when "%="

modAssign

when "negate"

negateExp

when "+SpecialRight"

addSpecialRight

when "\*SpecialRight"

multSpecialRight

when "verify"

verify

when "hear"

hearAction

when "say"

sayAction

when "goto"

gotoAction

when "gotoT"

gotoTrueAction

when "gotoF"

gotoFalseAction

when "param"

paramAction

when "gosub"

gosubAction

when "SEND\_ATTR"

sendAttributeAction

when "return"

returnAction

# Ends the program's execution.

when "terminate"

abort

# Indicates that there is no action for the current quadruple.

else

puts "falta cuadruplo " + quadruple[0]

end

end

# Method that starts the execution of the program. It shows a message

# so the programmer knows he is in the execution. It starts the

# execution of the quadruples one by one until the quadruple vector's

# total size is met. The counter increments by 1 to get the next

# quadruple.

def execution()

puts "Executing program...";

while @quadruplePointer < @quadrupleVector.count() do

executeQuadruple(@quadrupleVector[@quadruplePointer])

@quadruplePointer += 1

end

end

end

# Method that corresponds to the %= action.

# It assigns the value of a mod operation to a variable.

# It retrieves both the variable being assigned and the variable

# where the value will be assigned and it stores the result in a

# temporary variable in memory.

def modAssign()

op1 = @quadrupleVector[@quadruplePointer][2]

res = @quadrupleVector[@quadruplePointer][3]

# Get the value from op1 and store it in a temporary variable

temp = (obtainData(res) % obtainData(op1))

# Store the value of op1 in the @memory address of res

store(res, temp)

end

# Method that corresponds to the goto action.

# It gets the address of the place you need to

# jump to.

def gotoAction()

@quadruplePointer = @quadrupleVector[@quadruplePointer][3] - 1

end

# Method that corresponds to the gotoT action.

# It gets the address of the place you need to go to

# if the variable value is equal to true.

def gotoTrueAction()

value = obtainData(@quadrupleVector[@quadruplePointer][1])

@quadruplePointer = @quadrupleVector[@quadruplePointer][3] - 1 if value

end

# Method that corresponds to the gotoF action.

# It gets the address of the place you need to go to

# if the variable value is equal to false.

def gotoFalseAction()

value = obtainData(@quadrupleVector[@quadruplePointer][1])

@quadruplePointer = @quadrupleVector[@quadruplePointer][3] - 1 unless value

end