



Tecnológico de Monterrey

COVID19--

Diseño de Compiladores
3 de Junio 2020

Equipo 31
Los Contagiados por la Pandemia

José Arturo Villalobos
A00818214

Rodrigo Valencia
A00818256

Índice

Descripción del Proyecto	2
Propósito, Objetivos y Alcance	2
Análisis de Requerimientos y Casos de Uso Generales	2
Casos de Prueba	2
Proceso de Desarrollo	3
Lista de Commits	3
Bitácoras	21
Reflexión José	26
Reflexión Rodrigo	26
Descripción del Lenguaje	27
Nombre del Lenguaje: COVID19--	27
Principales características del lenguaje	27
Errores	27
Descripción del Compilador	29
Ambiente de Desarrollo	29
Análisis de Léxico	29
Análisis Sintáctico	30
Código intermedio y Análisis Semántico	33
Código de Operación y direcciones virtuales	33
Diagramas de Sintaxis	34
Acciones semánticas	38
Tabla de consideraciones semánticas	42
Administración de Memoria	43
Descripción de la Máquina Virtual	44
Ambiente de Desarrollo	44
Administración de Memoria	44
Direcciones de Memoria	45
Pruebas de Funcionamiento	46
Documentación	54
Compilación	54
Máquina Virtual	68
Manual de Usuario	81

1. Descripción del Proyecto

1.1. Propósito, Objetivos y Alcance

Esta versión del proyecto permitirá trabajar y manipular estadísticamente los datos contenidos en un archivo de texto. La carga de dicho archivo permitirá reconocer e interpretar cada encabezado del archivo como una variable. El objetivo de este proyecto es crear un lenguaje con una sintaxis amigable, fácil de aprender, muy bien estructurado para que cualquiera pueda utilizarlo.

1.2. Análisis de Requerimientos y Casos de Uso Generales

Este proyecto debe incluir una función que permita generar un arreglo con las variables detectadas en el archivo. Adicionalmente, el lenguaje debe proporcionar un conjunto de funciones estadísticas básicas que se puedan aplicar sobre ese dataframe.

También debe considerar algunas funciones que permitan correlacionar las variables presentes en el archivo.

Debe incluir al menos un par de funciones que permitan graficar y visualizar los datos.

Además, nuestro lenguaje debe manejar correctamente los elementos de :

- Recursión
- Funciones especiales estadísticas
- Arreglos
- Matrices
- Ciclos
- Condicionales
- Evaluación de expresiones
- Importar archivos externos
- Operaciones aritméticas
- Declaración de variables
- Asignación de valores a variables
- Declaración de funciones
- Llamadas a funciones

1.3. Casos de Prueba

Los casos de prueba que se utilizarán para probar el lenguaje serán:

- Factorial en Ciclos
- Factorial Recursivo
- Fibonacci en Ciclos

- Fibonacci Recursivo
- Ordena
- Encuentra
- Moda
- Mediana
- Varianza
- Coeficiente de correlación
- Histograma
- Plot Line
- Carga de Archivos
- Manejo de arreglos
- Manejo de matrices

1.4. Proceso de Desarrollo

1.4.1. Lista de Commits

Commits on Jun 3, 2020

Update README.md



rodvama committed 3 minutes ago

Verified [071b98f](#)

Update README.md



rodvama committed 8 minutes ago

Verified [ff10bf0](#)

Update README.md



rodvama committed 9 minutes ago

Verified [5e31d5a](#)

Update README.md



rodvama committed 12 minutes ago

Verified [288441e](#)

Update README.md



rodvama committed 16 minutes ago

Verified [cb73c6f](#)

Update README.md



rodvama committed 17 minutes ago

Verified [87cede8](#)

Commit Final



joseavc committed 24 minutes ago

[6d1d7ba](#)

Correccion para que funcione recursividad



joseavc committed 2 hours ago

[9e8c262](#)

Merge pull request [#32](#) from rodvama/rvm-pruebasFinales ...



rodvama committed 8 hours ago

Verified [7c4f412](#)

Arreglos finales



rodvama committed 8 hours ago

[520a267](#)

Commits on Jun 2, 2020

Encontrar, ordenar listos



rodvama committed 10 hours ago

[7dac67a](#)

merge master



rodvama committed 11 hours ago

[8f2c801](#)

Merge pull request [#31](#) from rodvama/jv-ordena ...



rodvama committed 11 hours ago

Verified [e50354c](#)

Guardar cambios



rodvama committed 11 hours ago

[5cc3473](#)

Correcion cuadruplo funcion encuentra



joseavc committed 11 hours ago

[bebe751](#)

Agregue el tamano del arreglo en el cuadruplo de encuentra



joseavc committed 11 hours ago

[1a0d116](#)

merge master



rodvama committed 13 hours ago

[bb21f9c](#)

Merge pull request [#30](#) from rodvama/jv-ordena ...



rodvama committed 13 hours ago

Verified [2c65939](#)

Generacion de codigo de funcion de Encuentra



joseavc committed 13 hours ago

[1e10c06](#)

merge master



rodvama committed 14 hours ago

[a1105f1](#)

Merge pull request [#29](#) from rodvama/jv-ordena ...



rodvama committed 15 hours ago

Verified [0dea532](#)

nuevas pruebas



rodvama committed 15 hours ago

[4816a84](#)

Generacion de la FunEsp Ordena y su cuadruplo



joseavc committed 15 hours ago

[58994f9](#)

Merge pull request [#28](#) from rodvama/rvm-MaVir ...



rodvama committed yesterday

Verified [42c3e58](#)

Histograma y plotline, guardar imagen en caso de no ser impresa en te... ...



rodvama committed yesterday

[51c08b1](#)

Commits on Jun 1, 2020

Histograma y plotline listo, faltan ultimas pruebas



rodvama committed yesterday

[9f10063](#)

Guardar cambios, se terminó funciones especiales y falta funciones pa... ...



rodvama committed 2 days ago

[ab45c21](#)

Arreglar conflictos



rodvama committed 2 days ago

[26702ee](#)

Guardar cambios para merge master: Moda, Mediana, Varianza Listo



rodvama committed 2 days ago

[da76a76](#)

Merge pull request [#27](#) from rodvama/jv-correcciones ...



rodvama committed 2 days ago

Verified [fcc6bf8](#)

Modificacion de funcion especial correlacion



joseavc committed 2 days ago

[e2241ae](#)

Merge pull request [#26](#) from rodvama/jv-correcciones ...



rodvama committed 2 days ago

Verified [765ec02](#)

Correccion de error de dataframes



joseavc committed 2 days ago

[e9b42b5](#)

Commits on May 31, 2020

Conflictos Resueltos



rodvama committed 3 days ago

[80680aa](#)

Merge pull request [#25](#) from rodvama/jv-funEsp ...



rodvama committed 3 days ago

Verified [cff232f](#)

Desarrollo de cuatruplos de Funciones especiales



joseavc committed 3 days ago

[7505983](#)

Resolver Conflictos



rodvama committed 3 days ago

[99b8b1e](#)

Guardar Cambios - se arregló archivos para aceptar arreglos bidimensi... ...



rodvama committed 3 days ago

[af17349](#)

Merge pull request [#24](#) from rodvama/jv-correcciones ...



joseavc committed 3 days ago

Verified [3f3419e](#)

Guardar Cambios - se arregló el guardar constantes declaradas en func... ...



rodvama committed 3 days ago

[1c49ae7](#)

Correccion en cubo semantico para que aceptara asignacion de string c... ...



joseavc committed 3 days ago

[b9ef61b](#)

Commits on May 31, 2020

Merge branch 'master' of <https://github.com/rodvama/compilers-final> i... ...



rodvama committed 3 days ago

[192c786](#)

Merge pull request [#23](#) from rodvama/jv-correcciones ...



rodvama committed 3 days ago

Verified [c6c6e34](#)

Correccion de cuadruplo de dataframe



joseavc committed 3 days ago

[ea94628](#)

Merge Master



rodvama committed 3 days ago

[0714599](#)

Merge pull request [#22](#) from rodvama/jv-correcciones ...



rodvama committed 3 days ago

Verified [7d87be6](#)

Guardar avances para hacer merge



rodvama committed 3 days ago

[8e4ea35](#)

Commits on May 30, 2020

Generacion de cuadruplos de Funciones Especiales que regresan valor (... ...)



joseavc committed 4 days ago

[8f7c25c](#)

Generacion de cuadruplo para Cargar Archivo



joseavc committed 4 days ago

[15847ee](#)

Generacion de cuadruplo de Dataframe



joseavc committed 4 days ago

[e7ef681](#)

Commits on May 29, 2020

Merge master Done



rodvama committed 5 days ago

[29cfb9e](#)

Guardar cambios, para hacer merge con master



rodvama committed 5 days ago

[8a360ed](#)

Merge pull request [#21](#) from rodvama/jv-correcciones ...



rodvama committed 5 days ago

Verified [259de14](#)

Commits on May 28, 2020

Corrección y adecuación ...



joseavc committed 5 days ago

[f227b40](#)

Commits on May 26, 2020

Se arregló conflictos después de merge con master



rodvama committed 8 days ago

[0811218](#)

Guardar cambios para hacer merge con master



rodvama committed 8 days ago

[a2ebcbe](#)

Commits on May 25, 2020

Merge pull request [#20](#) from rodvama/jv-Arreglos ...



rodvama committed 9 days ago

Verified [7cce494](#)

Commits on May 24, 2020

Generación de Código de Arreglos ...



joseavc committed 9 days ago

[4a01d56](#)

Generacion codigo Arreglos Access completo. No acepta arreglos como i... ...



joseavc committed 10 days ago

[2fdb90](#)

Pruebas a Arreglos



joseavc committed 10 days ago

[c6c10ed](#)

Generacion codigo Array Access. Sin Terminar



joseavc committed 10 days ago

[d0c9e5d](#)

Commits on May 23, 2020

Generacion de codigo de declaracion de Arreglos



joseavc committed 11 days ago

[9e90dcb](#)

Commits on May 22, 2020

Se agrego, un ambiente virtual, para manejar paquetes



rodvama committed 12 days ago

[3398394](#)

Commits on May 21, 2020

Maquina virtual, operaciones listo, falta funciones especiales y coma... ...



rodvama committed 13 days ago

[7235a06](#)

Commits on May 18, 2020

Maquina Virtual - v1



rodvama committed 15 days ago

[6bdf9ac](#)

Merge pull request [#19](#) from rodvama/rvm-vm ...



rodvama committed 16 days ago

Verified [f8a4443](#)

clase memoria virtual - v2



rodvama committed 16 days ago

[ef91a71](#)

clase memoria virtual - v1



rodvama committed 16 days ago

[7afd90f](#)

Quick Manual Reference -v2



rodvama committed 16 days ago

[da1608a](#)

Quick Manual Reference



rodvama committed 16 days ago

[bbfb4c1](#)

Commits on May 11, 2020

Merge pull request [#17](#) from rodvama/jv-GenCodFunciones ...



joseavc committed 23 days ago

Verified [dc0dc8c](#)

Corrección de Errores en Modulos ...



joseavc committed 23 days ago

[49d448f](#)

Merge pull request [#16](#) from rodvama/jv-iniciarMemoria ...



joseavc committed 23 days ago

Verified [db04dfd](#)

Manejo de Memoria ...



joseavc committed 23 days ago

[f9f43a2](#)

Detectar funciones especiales y actualizar dimensiones



joseavc committed 23 days ago

[613ae5a](#)

Se arreglo el problema del estatuto de retorno



joseavc committed 23 days ago

[f505087](#)

Commits on May 10, 2020

Corrigiendo errores. Todavia tiene fallas en el Retorno



joseavc committed 24 days ago

[c9c6225](#)

Commits on May 9, 2020

Merge pull request [#15](#) from rodvama/jv-funcDefinition ...



joseavc committed 25 days ago

Verified [3cab61e](#)

Código Intermedio para Modules Definition y Module Call ...



joseavc committed 25 days ago

[20fe48d](#)

Merge pull request [#14](#) from rodvama/jv-updateDirFunc ...



joseavc committed 25 days ago

Verified [82cf1c6](#)

Se agrego el atributo de cantidad de cuádruplos en el Directorio de F... ...



joseavc committed 25 days ago

[4f3312d](#)

Ajustes2



joseavc committed 25 days ago

[e4b425e](#)

Commits on May 8, 2020

Merge pull request [#13](#) from rodvama/ajustes ...



joseavc committed 26 days ago

Verified [94981d2](#)

Ajustes



joseavc committed 26 days ago

[f4590ed](#)

Commits on May 6, 2020

Merge pull request [#12](#) from rodvama/jv-ctesNegativos ...



joseavc committed 27 days ago

Verified [17142af](#)

Operaciones con Constantes y Negativos ...



joseavc committed 27 days ago

[470305a](#)

Commits on May 2, 2020

Merge pull request [#11](#) from rodvama/jv-GC-Condicionales ...



joseavc committed on May 2

Verified [b2aa12f](#)

Merge pull request [#10](#) from rodvama/jv-ciclos ...



joseavc committed on May 2

Verified [99232d1](#)

Generación de Código estatutos Condicionales - While y For ...



joseavc committed on May 2

[fe53e72](#)

Merge pull request [#9](#) from rodvama/jv-constantes ...



joseavc committed on May 2

Verified [304511f](#)

Añadir constantes a Pila de Operandos ...



joseavc committed on May 2

[c7e7939](#)

Merge pull request [#8](#) from rodvama/jv-if ...



joseavc committed on May 2

Verified [d723da8](#)

Comentarios



joseavc committed on May 2

[e9a9e75](#)

Generación de Código - Si Sino (If Else) ...



joseavc committed on May 2

[f78ac7f](#)

Commits on Apr 27, 2020

Merge pull request [#7](#) from rodvama/jv-semanticExp ...



joseavc committed on Apr 27

Verified [a56b24b](#)

Generación de Código de Estatutos Secuenciales ...



joseavc committed on Apr 27

[3bcfb4f](#)

Commits on Apr 26, 2020

Generación de Código de Expresiones Aritméticas ...



joseavc committed on Apr 26

[f8d8e5b](#)

Deteccion de columnas y renglones en la declaracion de variables con



joseavc committed on Apr 26

[95ae378](#)

Se arreglo prints en archivo



rodrigo committed on Apr 26

[8634996](#)

Merge pull request [#6](#) from rodvama/jv-cubo ...



joseavc committed on Apr 26

Verified [d3f21ca](#)

Merge pull request [#5](#) from rodvama/jv-dirFunc ...



joseavc committed on Apr 26

Verified [33cfc77](#)

Commits on Apr 25, 2020

Cubo Semantico ...



joseavc committed on Apr 25

[7b3479c](#)

Puntos Neurálgicos de DirFunc ...



joseavc committed on Apr 25

[6a7a2be](#)

Commits on Apr 21, 2020

Directorio de funciones y Tabla de Variables ...



joseavc committed on Apr 21

[dd9c3be](#)

Commits on Apr 18, 2020

Error corregido: en la gramatica de funcion no era repetido. Se modif... ...



joseavc committed on Apr 18

[95dfa8d](#)

Commits on Apr 12, 2020

Merge pull request [#4](#) from rodvama/rvm-filefunction ...



rodvama committed on Apr 12

Verified [8ae4e70](#)

Agregue función para leer archivos, se arreglo la función de lectura ...



rodrigo committed on Apr 12

[b42e7e4](#)

Merge pull request [#3](#) from rodvama/jv-par ...



rodvama committed on Apr 12

Verified [8730ae8](#)

Tokens de Strings y Chars completados y probados



joseavc committed on Apr 12

[0d9cb68](#)

Commits on Apr 11, 2020

Pendiente las ER de String y Char



joseavc committed on Apr 11

[9918ced](#)

Parser Completado ...



joseavc committed on Apr 11

[ab30cf1](#)

Commits on Apr 10, 2020

Parser parte 1 ...



joseavc committed on Apr 10

[2e1464d](#)

Commits on Apr 5, 2020

Merge pull request [#2](#) from rodvama/jv-lex ...



rodvama committed on Apr 5

Verified [4468d83](#)

Analizador Léxico y PLY ...



joseavc committed on Apr 5

[4abe2f9](#)

Rename lex.py.py to lex.py



rodvama committed on Apr 5

Verified [35b1a85](#)

Update README.md



rodvama committed on Apr 5

Verified [201fc0b](#)

Merge pull request [#1](#) from rodvama/rvm-lex ...



rodvama committed on Apr 5

Verified [04ded5c](#)

First declarations of TOKENS



rodrigo committed on Apr 5

[41dfc40](#)

Commits on Mar 27, 2020

Update README.md



rodvama committed on Mar 27

Verified [b1992d7](#)

Initial commit



rodvama committed on Mar 27

Verified [a1ec66e](#)

1.4.2. Bitácoras

Bitácora 1 - 12 de Abril del 2020 (Avance 1)

- Se definieron los tokens y su gramática formal para cada uno.
- Se terminó los diagramas de léxico, para todo el programa.
- Además se implementaron dichos diagramas en los archivos de léxico y de sintaxis ya probados con ejemplos dentro de archivos.
- Se corrieron pruebas para comprobar el funcionamiento de dichos programas.

¿Funciona?

Sí

¿Faltó una parte?

No

¿Corrieron suficientes test-cases?

Solo uno

¿Tienen todavía ambigüedades en la gramática?

Fueron arregladas

¿Pueden marcar errores particulares o solo uno genérico (syntax error)?, etc
Sí, ya que podemos imprimir la línea exacta del código que tienen error

Bitácora 2 - 20 de Abril del 2020 (Avance 2)

Directorio de procedimientos y Tabla de Variables

- Se crearon las clases de DirFunc y TablaVars que representan el directorio de procedimientos y tabla de variables respectivamente
- Se implementaron sus funciones básicas
- Se hicieron pocas pruebas
- Pendiente de hacer pruebas más exhaustivas
- Pendiente de implementar los puntos neurálgicos en el parser.

Bitácora 20 - 27 de Abril del 2020 (Avance 3)

- Se hizo el cubo semántico.
- Se hizo el cubo semántico para funciones especiales
- Se agregaron puntos neurálgicos faltantes del avance anterior.
Específicamente para poder agregar las funciones y variables al directorio de funciones y tabla de variables respectivamente.
- Se actualizó el directorio de funciones, para aceptar arreglos.
- Se agregó generación de código de expresiones aritméticas.
- Ya se probaron, demostrando que compila y funciona.
- Se definieron los puntos neurálgicos de los estatutos secuenciales de "lectura" y "escritura", así como la implementación de los puntos neurálgicos correspondientes en la gramática.
- Se hicieron pruebas de funcionamiento y pasaron.
- Avance 2 y 3 Completo.

Bitácora 28 abril - 4 Mayo (Avance 4)

- Generación de Código - Si Sino (If Else)
 - Se crearon la pila de Saltos y el arreglo que va a contener la lista de cuádruplos.
 - Siguiendo las hojas de apoyo de Elda, se agregaron los puntos neurálgicos a los estatutos de 'Si' (if) y 'Sino' (else). Se crearon e implementaron las funciones que representan estos puntos neurálgicos.
 - Se definió una función (nextQuad) que nos regresa el tamaño actual del arreglo de cuádruplos. Esta función sirve para obtener el próximo o anterior índice del arreglo de cuádruplos para ponerlos en los cuádruplos de GOTO.
 - Además, se decidió implementar funciones de impresión de errores, para tener un control de los tipos de errores que va a manejar el lenguaje. Por lo pronto, solo se definieron dos tipos de errores: Type Mismatch y Out of Bounds.

- **Añadir constantes a Pila de Operandos**
 - Se definieron puntos neurálgicos para incluir constantes dentro de la pila de Operandos.
- **Generación de Código estatutos Condicionales - While y For**
 - Se incluyeron los puntos neurálgicos en las reglas sintácticas de los estatutos condicionales "desde"(for) y "mientras"(while).
 - Así como también se implementaron las funciones que representan dichos puntos neurálgicos.
 - Se hicieron pruebas y funcionan perfectamente.
 - Queda pendiente la aceptación de números negativos.

Bitácora 5 Mayo - 11 Mayo (Avance 5)

- Se agregó el atributo de cantidad de cuádruplos en el Directorio de Funciones
- Código Intermedio para Modules Definition y Module Call
 - Se implementaron los puntos neurálgicos para la generación de código en Declaración de funciones y llamadas a funciones. A falta de pruebas, funciona correctamente.
 - En el directorio de funciones se agregó la función ListaTipos.
- Se arreglo el problema del estatuto de retorno
- Detectar funciones especiales y actualizar dimensiones
- Corrección de Errores en Módulos
 - Se corrigió el error que había cuando se llamaba una función con return. Se le agregó un cuádruplo que guardaba el resultado de la función en un Temporal y lo mete a la pila de Operandos.
- Manejo de Memoria
 - Se agregó la funcionalidad de buscar la posición de memoria virtual en la tabla de variables, desde el directorio de funciones.
 - Se modificó gran parte del parser para aceptar los nuevos atributos semánticos del Directorio de funciones y la Tabla de Variables.
- Generación de Código de Funciones Funcionando
 - Se probó el ejemplo del programa Patito de la hoja de Excel de Elda, y los resultados fueron exactamente iguales.

Bitácora 11 de mayo - 18 de Mayo (Avance 6)

- Se planeó como se maneja la memoria en el programa, que será a partir de una pila.
- Se desarrolló una clase de memoria virtual, que maneja las direcciones para el programa.
 - Función para guardar valor
 - Función para obtener valor
 - Función que permite sacar la siguiente dirección
 - Función para imprimir la memoria completaQueda pendiente, agregar funciones que se pueden usar.
- Se probó la clase de memoria virtual, y se empezó a evaluar cómo se incorporará con la máquina virtual.
- Se inició implementación de la máquina virtual.

- Se enfrentaron problemas al probar la máquina virtual, en las funciones para manejar la memoria virtual y lograr hacer la suma y restas.

Bitácora 18 de mayo - 22 de Mayo (Avance 7)

- Se trabajó en la implementación de los arreglos
- Se generaron los cuádruplos para la generación de código para declarar Arreglos y Matrices.
- Se generaron los cuádruplos para la generación de código para acceder a los Arreglos y Matrices.
- Se corrigió el error que no permitía indexar un Arreglo en el índice de otro arreglo. Esto se corrigió implementando correctamente la pila de Dimensiones.
- Falta arreglar errores en máquina virtual, para acceder a la tabla de variables.
- Falta ajustar máquina virtual para leer cuádruplos de arreglos
- Pruebas de integración
- Se empezó la documentación final

Bitácora 18 de mayo - 22 de Mayo (Avance 7)

- Se trabajó en la implementación de los arreglos
- Se generaron los cuádruplos para la generación de código para declarar Arreglos y Matrices.
- Se generaron los cuádruplos para la generación de código para acceder a los Arreglos y Matrices.
- Se corrigió el error que no permitía indexar un Arreglo en el índice de otro arreglo. Esto se corrigió implementando correctamente la pila de Dimensiones.
- Falta arreglar errores en máquina virtual, para acceder a la tabla de variables.
- Falta ajustar máquina virtual para leer cuádruplos de arreglos
- Pruebas de integración
- Se empezó la documentación final

Bitácora 23 de mayo - 31 Mayo (Avance 8)

- **Se ajustó el parser para mandar cuádruplos correctamente**
- **Se agregó al parser la creación de cuádruplos para funciones especiales**
- Se hicieron varios cambios a la máquina virtual
 - Se adaptó para guardar constantes primero antes que nada.
 - Ya soporta arreglos dimensionales
 - Ya soporta arreglos bidimensionales
 - Se adecuó para soportar carga de archivos
 - Se trabajó en funciones especiales

Falta terminar funciones especiales y ajustar el dataframe

Falta un 80% de la documentación

1.4.3. Reflexión José


Ha sido el proyecto más retador que he tenido en la carrera. Desde semestres anteriores se escuchaban los rumores de lo que me esperaba en esta materia tan "temida" entre los ITC. Como a mi no me gusta dejar todo al final, desde antes de iniciar el semestre me comencé a preparar para llevar este curso. El primer día de clases que escogimos nuestra pareja, le dije a mi compañero Rodrigo que NO íbamos a dejar nada para el final. Que todo lo íbamos a ir entregando en tiempo y forma porque al final del semestre no estaba dispuesto a olvidar el resto de mis materias para terminar este proyecto. Y en base a eso, fue como trabajamos en todo el semestre. Hacíamos todo a tiempo y forma, no nos retrasamos ni una vez. Aunque en los primeros avances, al todavía no tener claro todo el proyecto, batallamos mucho para iniciar los primeros puntos neurálgicos. Pero ya después de declarar y desarrollar los primeros, el resto fue mucho más fácil. Siguiendo los algoritmos de Elda para desarrollar la mayoría de la generación de código se nos simplificó y facilitó mucho el desarrollo. Me siento orgulloso de lo que hicimos, no nos estresamos más de lo necesario porque trabajamos el proyecto durante todo el semestre sin retrasos. Y definitivamente, se aprende mucho mejor y se asimilan mejor los conceptos vistos en clase, llevándolos a la práctica con un proyecto de gran exigencia donde te pida utilizar toda la teoría vista en clase. A diferencia de la mayoría de los ITCs que dicen rumores negativos de este curso, YO me llevo una gran experiencia, y me atrevo a decir que ha sido uno de los mejores cursos que he llevado y, con Elda, una de las mejores maestras que he tenido. Elda, Muchas Gracias por retornos y exigirnos superar nuestros propios límites mentales en este curso.


José Arturo Villalobos

1.4.4. Reflexión Rodrigo

Gracias a la situación mundial actual, el desarrollo del proyecto, ha sido un reto, sobre todo, la comunicación de equipo, ya que no fue tarea fácil el ponerse de acuerdo por los diferentes horarios, para asignarse tareas, no existía el juntarse para desarrollar las ideas en conjunto, causando que fuera difícil expresar y dejar las ideas en claro el uno con el otro, aunque tengo que aceptar, que con el paso del desarrollo del proyecto, fuimos descubriendo cómo comunicarnos de manera más eficiente, desde generando llamadas para discutir qué haríamos, hasta mandarnos videos por whatsapp para enseñarnos los errores del otro de manera, de que el otro pudiera revisarlo en su tiempo libre, además en sí el proyecto es complejo, debido a que se ven muchos temas, que un principio parecían complicados, pero conforme va pasando el semestre, van agarrando sentido y más en el proyecto, el cual es la

culminación de todos ellos. Esta clase siendo de las últimas de la carrera, es de las más completas que hay que uno sino es que todos, pero si la mayoría de los conocimientos que se aprenden a través de la carrera, como algoritmos, sobre todo estructura de datos, hasta incluso administración de proyectos, ya que se debe de llevar un control de todo el desarrollo.



Rodrigo Valencia Maciel

2. Descripción del Lenguaje

2.1. Nombre del Lenguaje: COVID19--

2.2. Principales características del lenguaje

Es un lenguaje básico que cubre con el uso de diferentes tipos de variables (int, float, string, char, dataframes), además de manejar arreglos unidimensionales y bidimensionales, así como dataframes que serán registrados a través de la carga de archivos.

2.3. Errores

No soporta archivos de dataframes con matrices, simplemente es un arreglo.

Errores en fase de compilación:

- Caracter ilegal '%' at '%'
- Error de Sintaxis
- Error de Sintaxis en EOF
- Error Type Mismatch
- Error: Memoria llena; demasiadas '%' de tipo '%'
- Error: Tipo de variable no existente
- Error: Variable '%' no declarada
- Error: Operador no esperado
- Error: Operacion invalida
- Error: Funcion '%' ya existe
- Error: Función no existe
- Error: Parámetros incorrectos
- Error: Muchos argumentos
- Error: Mismatch de Argumentos

- Error: En funcion especial: {}
- Error: En funcion especial, el tipo del primer parámetro no es dataframe
- Error: En funcion especial, el tipo del primer parámetro no es arreglo int/float
- Error: En funcion especial correlaciona, los primeros dos parámetros no son dataframes.
- Error: En funcion especial correlaciona, los índices deben ser enteros
- Error: En funciones especiales void histograma, el tipo del segundo y tercer parámetro debe ser int.
- Error: En funciones especiales void histograma, el tipo del primer parámetro debe ser dataframe, int o float.
- Error: En funciones especiales void plotline, el tipo del tercer parámetro debe ser entero.
- Error: En funciones especiales void plotline, el tipo del primer y segundo parámetro debe ser dataframe o constante entera o flotante.
- Error: En funciones especiales void Ordena, el arreglo del parámetro no existe.
- Error en función Encuentra. El arreglo y el valor a encontrar deben ser del mismo tipo
- Error: No se pudo crear la variable: '%s' en la funcion: '%s'
- Error: Imposible actualizar parámetros de una funcion no existente: '%s'
- Error : Variable '%s' duplicada
- Error: Índice de un arreglo no es entero
- Error: No existe variable dimensionada
- Error: No se puede acceder al index porque la variable no es dimensionada
- Error: La variable no es matriz
- Error: La variable no es dimensionada
- Error: Al cargar un archivo, el tipo del segundo parámetro no es string
- Error: Al cargar un archivo, el tipo del primer parámetro no es dataframe
- Error: el tipo que intenta retornar no es correcto
- Error: Overflow

Errores en fase de ejecución:

- Error: No encontrar el valor deseado dentro de una dirección de memoria.
- Error: No detectar el tipo de variable dentro de una dirección de memoria.
- Error: Los datos de un dataframe no tienen mediana o se repite.
- Error: Los datos de un dataframe no tienen media o se repite.
- Error: Los datos de un dataframe no tienen moda o se repite.
- Error: Los datos de un dataframe no tienen varianza o se repite.
- Error: Los datos de dos dataframes no son del mismo tamaño al ejecutar el coeficiente de correlación.
- Error: Al no poder graficar un histograma con los datos de un dataframe.
- Error: Al no poder graficar un plot line con los datos de un dataframe.

- Error: Usar índices que no estén dentro de la dimensión del arreglo.
- Error: Usar índices que no estén dentro de la dimensión de un dataframe.
- Error: No proporcionar el path correcto para cargar un archivo a un dataframe.

3. Descripción del Compilador

3.1. Ambiente de Desarrollo

El desarrollo del proyecto se realizó en computadores con sistema operativo MacOS y Windows. Como lenguaje de programación para desarrollar el compilador y la máquina virtual fue Python3. Para la generación de Analizadores Léxico y Sintáctico se utilizaron las herramientas lex & yacc de PLY.

3.2. Análisis de Léxico

Los tokens del lenguaje son los siguientes:

Keywords:

- PROGRAM : 'programa'
- PRINCIPAL : 'principal'
- VAR : 'var'
- FUNCION : 'funcion'
- INT : 'int'
- FLOAT : 'float'
- STRING : 'string'
- CHAR : 'char'
- DATAFRAME : 'dataframe'
- VOID : 'void'
- SINO : 'sino'
- MIENTRAS : 'mientras'
- HAZ : 'haz'
- DESDE : 'desde'
- HASTA : 'hasta'
- HACER : 'hacer'
- VARIABLES : 'variables'
- HISTOGRAMA : 'histograma'
- PLOTLINE : 'plotline'
- MEDIA : 'media'
- MEDIANA : 'mediana'
- MODA : 'moda'
- VARIANZA : 'varianza'

- CORRELACIONA : 'correlaciona'
- ORDENA : 'ordena'
- ENCONTRAR : 'encontrar'
- REGRESA : 'regresa'
- LEE : 'lee'
- ESCRIBE : 'escribe'
- CARGA : 'carga'
- SI : 'si'
- ENTONCES : 'entonces'

Operadores

- ADD : '+'
- SUB : '-'
- MUL : '*'
- DIV : '/'
- OR : '|'
- AND : '&'
- LT : '<'
- LE : '<='
- GT : '>'
- GE : '>='
- EQUAL : '=='
- NOT_EQUAL : '!='

- ASSIGN: '='

- DOT: '.'

Separadores

- LPAREN: '('
- RPAREN: ')'
- LBRACE: '{'
- RBRACE: '}'
- LBRACK: '['
- RBRACK: ']'
- SEMICOLON: ';'
- COMMA: ','

Literals

- ID: `[a-zA-Z_][a-zA-Z_0-9]*`
- CTE_INT: `\d+`
- CTE_FLOAT: `\d+\.\d+`
- CTE_CH: `'[a-zA-Z]'`
- CTE_STR: `"(\\"([^\\""]|\\.)+\\")|'([^\''"]|\\.)+\\')"`
- COMMENT: `\%\%.`

3.3. Análisis Sintáctico

Gramática Formal empleada para representar las estructuras sintácticas:

S0 → PROGRAMA

PROGRAMA → *programa id* ; VAR' FUNC' PRINCIPAL

VAR' → VAR | empty

FUNC' → FUNCION FUNC' | empty

PRINCIPAL → *principal ()* BLOQUE

VAR → *var* VAR"

VAR" → TYPE : LISTA_IDS VAR""

VAR "" → VAR" | empty

TYPE → TIPO_SIMPLE | TIPO_COMPUESTO

FUNCION → *funcion* TIPO_FUN *id* (PARAMETROS) VAR' BLOQUE

TIPO_FUN → *void* | TIPO_SIMPLE

PARAMETROS → PARAM | empty

PARAM → TIPO_SIMPLE *id* PARAM'

PARAM' → , PARAM | empty

TIPO_SIMPLE → *int* | *float* | *char*

TIPO_COMPUESTO → *dataframe* | *string*

LISTA_IDS → LISTA ;

LISTA → *id* DD LISTA'

LISTA' → , LISTA | empty

DD → DIM_DEC | empty

DIM_DEC → [*cte_int*] DIM_DEC'

DIM_DEC' → [*cte_int*] | empty

DIM_INDEX → [EXP] DIM_INDEX'

DIM_INDEX' → [EXP] | empty

BLOQUE → { EST }

EST → ESTATUTOS EST | empty

ESTATUTOS → ASIGNACION | LLAMADA | RETORNO | LECTURA | ESCRITURA
| CARGA_DATOS | DECISION | CONDICIONAL | NO_CONDICIONAL |
FUNCIONES_ESPECIALES_VOID

ASIGNACION → VARIABLE = ASIG

ASIG → LLAMADA | EXP ;

VARIABLE → *id* DI

DI → DIM_INDEX | empty

RETORNO → *regresa* (EXP) ;

LECTURA → *lee* (LISTA_IDS) ;

ESCRITURA → *escribe* (ESC) ;

ESC → ESC' ESC"

ESC' → EXP

ESC" → , ESC | empty

CARGA_DATOS → *cargaArchivo* (*id* , *cte_string* , CA , CA) ;

CA → *id* | *cte_int*

DECISIÓN → *si* (EXPRESION) *entonces* BLOQUE *SINO*

SINO → *sino* BLOQUE | empty

CONDICIONAL → *mientras* (EXPRESION) *haz* BLOQUE

NO_CONDICIONAL → *desde* VARIABLE = EXP *hasta* EXP *hacer* BLOQUE

FUNCIONES_ESPECIALES_VOID → *Variables* (*id* , *id* , *id*) ; | *histograma* (*id* ,
cte_int , *cte_int*) ; | *plotline* (CA , CA , *cte_int*) ; | *ordena* (*id*) ;

FUNCIONES_ESPECIALES → *FE* (*id* , *cte_int* , *cte_int*) | *correlacion* (*id* , *id* ,
cte_int , *cte_int*) | *encuentra* (*id* , *var_cte*)

FE → *Media* | *Mediana* | *Moda* | *Varianza*

LLAMADA → *id* (LLAMADA') ;

LLAMADA' → EXP LLAMADA" | empty

LLAMADA" → , LLAMADA' | empty

VAR_CTE → *cte_str* | *cte_ch* | - VAR_NUM | VAR_NUM

VAR_NUM → *cte_int* | *cte_float*

EXPRESION → MEGA_EXP EXPRESION'

EXPRESION' → = EXPRESION | empty

MEGA_EXP → SUPER_EXP MEG

MEG → OP_R MEGA_EXP | empty

OP_L → & | |

SUPER_EXP → EXP SP

SP → OP_R EXP | empty

OP_R → < | > | <= | >= | != | ==

EXP → TERMINO EXP'

EXP' → OP_A EXP | empty

OP_A → + | -

TERMINO → FACTOR TERM

TERM → OP_A' TERMINO | empty

OP_A' → * | /

FACTOR → VAR_CTE | (EXP) | VARIABLE | LLAMADA |
FUNCIONES_ESPECIALES

3.5. Código intermedio y Análisis Semántico

3.5.1. Código de Operación y direcciones virtuales

Aritméticos y Lógicos

ADD : '+'

SUB : '-'

MUL : '*'

DIV : '/'

OR : '|'

AND : '&'

LT : '<'

LE : '<='

GT : '>'

GE : '>='

EQUAL : '=='

NOT_EQUAL : '!='

ASSIGN : '='

Saltos

GOTO : 'GOTO'

GOTOOF : 'GOTOOF'

Funciones

ESCRIBE : 'escribe'

LEE : 'lee'

REGRESA : 'regresa'

CARGA : 'carga'

HISTOGRAMA : 'histograma'

PLOTLINE: 'plotline'

MEDIA: 'media'

MODA : 'moda'

MEDIANA : 'mediana'

VARIANZA : 'varianza'

CORRELACION : 'correlacion'

ORDENA : 'ordena'

ENCONTRAR : 'encontrar'

ERA : 'ERA'

PARAM : 'PARAM'

GOSUB : 'GOSUB'

ENDPROC : 'ENDPROC'

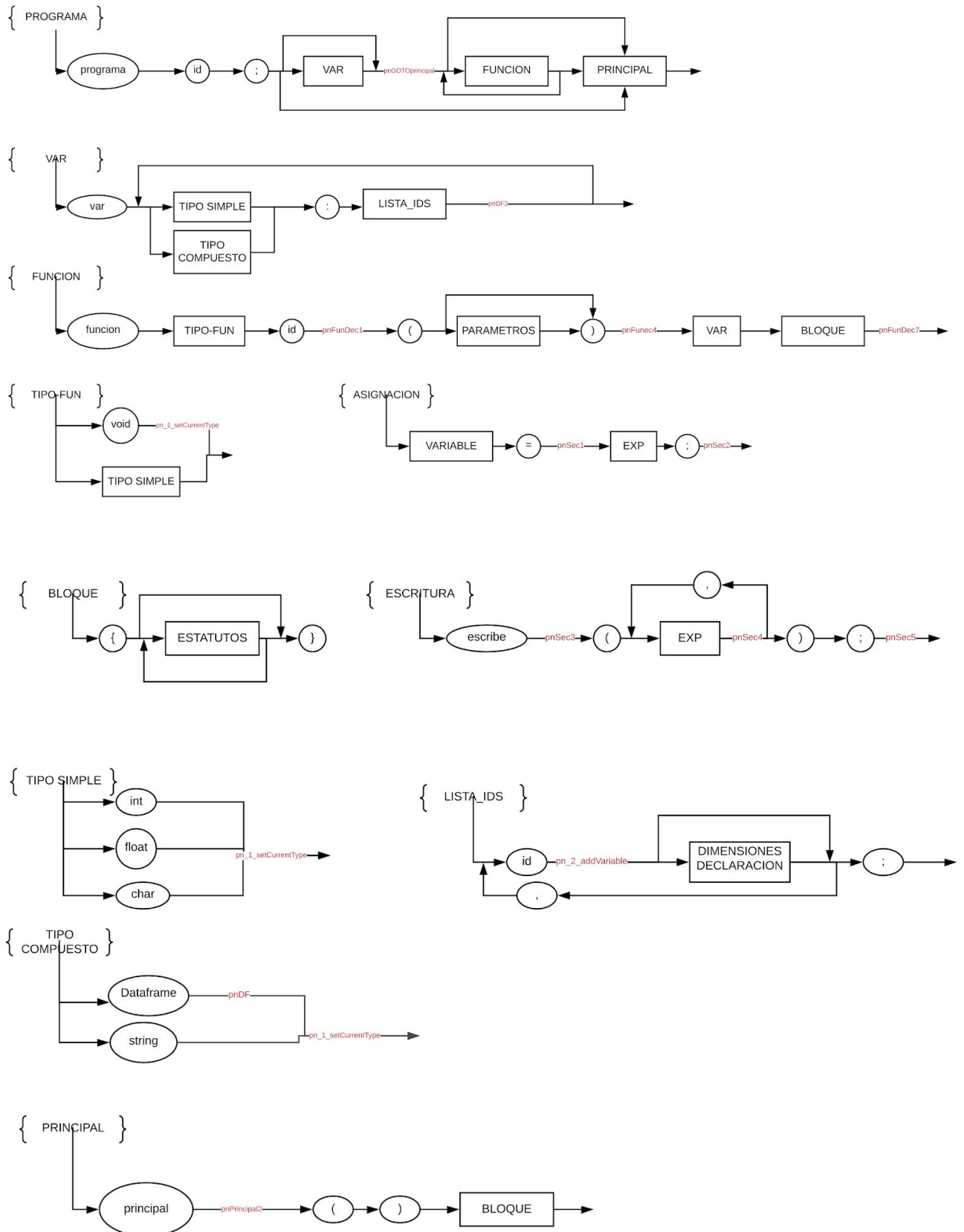
Arreglos

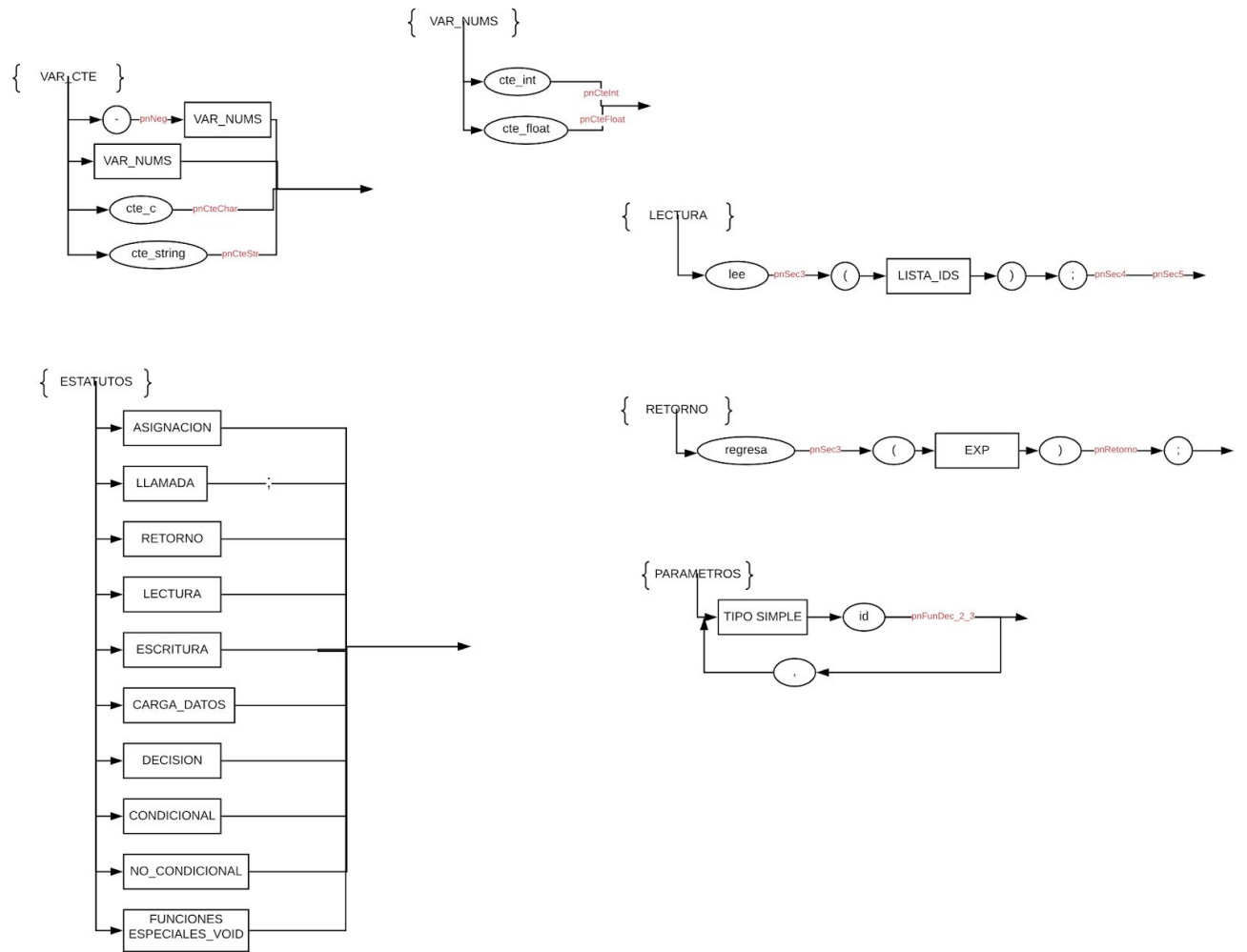
VERIFICA : 'VER'

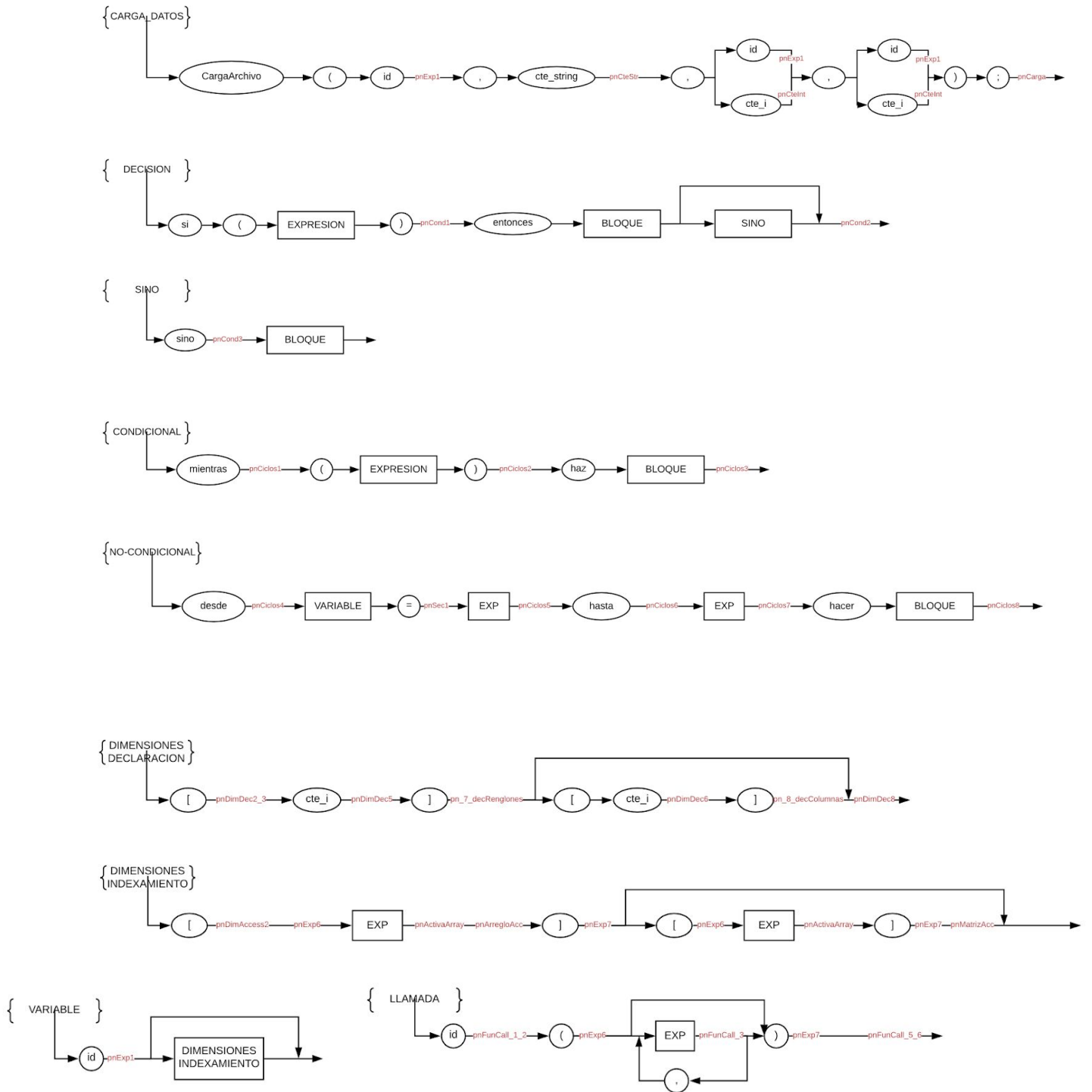
Programa

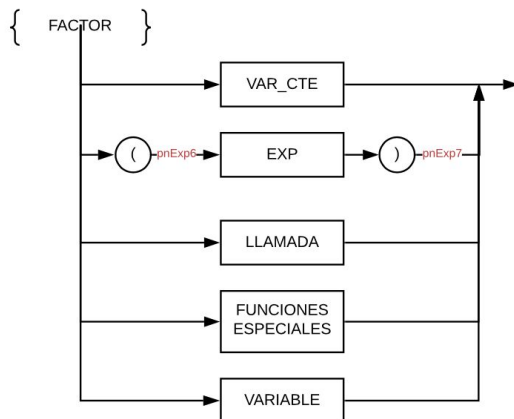
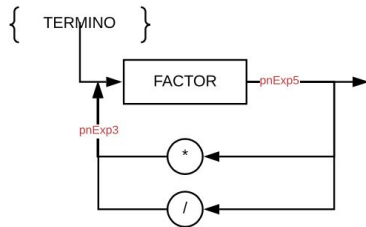
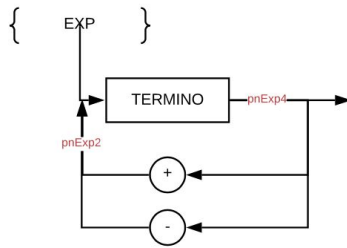
FINPROGRAMA : 'FINPROGRAMA'

3.5.2. Diagramas de Sintaxis









3.5.3. Acciones semánticas

Puntos neurálgicos incluidos en los diagramas anteriores:

Punto Neurálgico	Descripción
pnGOTOprincipal	Cuádruplo GOTO que va a la función principal
pnPrincipal2	Llena el cuádruplo pendiente GOTO indicando la dirección a la que tiene que ir el GOTO a principal.
pn_1_setCurrentType	Le asigna el tipo actual a la variable CurrentType
pn_2_addVariable	Agrega la variable a la Tabla de Variables
pnDF	Activa la variable booleana boolDataf cuando se está declarando un dataframe

pnDF2	Desactiva la variable booleana boolDataf cuando termina la declaracion de variables de tipo dataframe
pnFunDec1	Agrega el nombre de la función al Directorio de Funciones
pnFunDec_2_3	Cuenta la cantidad de parámetros que tiene una función y agrega dichos parámetros como variables locales de la función actual
pnFunDec4	Modifica la cantidad de parámetros de una función en el directorio de funciones
pnFunDec7	Elimina el directorio de funciones
pnDimDec2_3	Set id as an Array (isArray = true)
pnDimDec5	Guardar límite de Columnas
pnDimDec6	Guarda la cantidad de renglones (Significa que es matriz)
pnDimDec8	Actualiza el pointer de memoria tomando los espacios necesarios para el arreglo
pn_7_decRenglones	Lee la cantidad de renglones que hay y la guarda en una variable
pn_8_decColumnas	Lee la cantidad de columnas que hay y la guarda en una variable
pnDimAccess2	acceder a Arreglos
pnActivaArray	Activa el bool isArray para indicar que la variable es un arreglo
pnArregloAcc	Acceder al índice del arreglo
pnMatrizAcc	Acceder índice de matriz
pnSec1	Mete '=' a la pila operadores
pnSec2	Checa si en el top de la pila de operadores hay una asignación (=)
pnSec3	Meter escribir o leer o regresa a la pila
pnSec4	Checa si el top de la pila de operadores es lectura o escritura o retorno

pnSec5	Hace pop a la pila de operadores
pnFunCall_1_2	Verifica que la función exista en el directorio de funciones
pnFunCall_3	Va guardando los parámetros para después revisarlos
pnFunCall_5_6	Genera el cuádruplo GOSUB
pnFunEsp1	Identifica el nombre de la función especial y lo mete a la pila de operandos
pnFunEsp2	Generación de cuádruplo de funciones especiales que regresan valor
pnFunEsp3	Función para manejar la función especial correlación y generar sus cuádruplos
pnFunEsp4	Función para manejar la función especial encuentra y generar sus cuádruplos
pnFunEspVoid1	Función para generar cuádruplos de funciones especiales void que grafican
pnFunEspVoid2	Función para manejar y generar cuádruplos para la función especial ordena
pnCarga	Generación del cuádruplo de Carga de Archivos
pnRetorno	Cuádruplo que genera el retorno de una función
pnCond1	Genera el cuádruplo GOTOF en la condición SI (if) después de recibir el booleano generado por la expresión
pnCond2	Rellena el cuádruplo para saber cuando terminar la condición
pnCond3	Genera el cuádruplo GOTO para SINO (else) y completa el cuádruplo
pnCiclos1	Mete el siguiente cuádruplo a pSaltos. Que representa la ubicación a donde regresará al final del ciclo para volver a evaluar la condición
pnCiclos2	Genera el cuádruplo de GOTOF
pnCiclos3	Genera el cuádruplo GOTO para regresar al inicio del ciclo y volver evaluar la nueva condición. Aquí también se rellena el GOTOF anterior

pnCiclos4	Activa la variable bool de ForBool para indicar que está entrando a un For
pnCiclos5	Hace verificaciones si existen las variables y si los tipos son compatibles
pnCiclos6	Hace push de la variable que haya en el For
pnCiclos7	Genera el cuádruplo GOTO para el For
pnCiclos8	Genera el cuádruplo GOTO para el For
pnCteStr	Agrega la constante String a la pila de Constantes
pnCteInt	Agrega la constante Entera a la pila de Constantes
pnCteChar	Agrega la constante Char a la pila de Constantes
pnCteFloat	Agrega la constante Flotante a la pila de Constantes
pnNeg	Si hay un negativo, activa el booleano para indicar que es negativo
pnExp1	Añade id y Tipo a poper y pTipo respectivamente
pnExp2	Añade + o - al POper
pnExp3	Añade * o / al POper
pnExp4	Checa si el top del POper es un + o - para generar el cuádruplo de esa operación
pnExp5	Checa si el top de la pila de operadores es un * o / para generar el cuádruplo
pnExp6	Agrega fondo falso
pnExp7	Quita fondo falso
pnExp8	Mete un operador relacional a la pila de operadores
pnExp9	Verifica si el top de la pila de operadores es un operador relacional para generar el cuádruplo de operación
pnExp10	Meter un operador lógico a pila de operadores
pnExp11	Checa si el top de la pila de operadores es un operador lógico

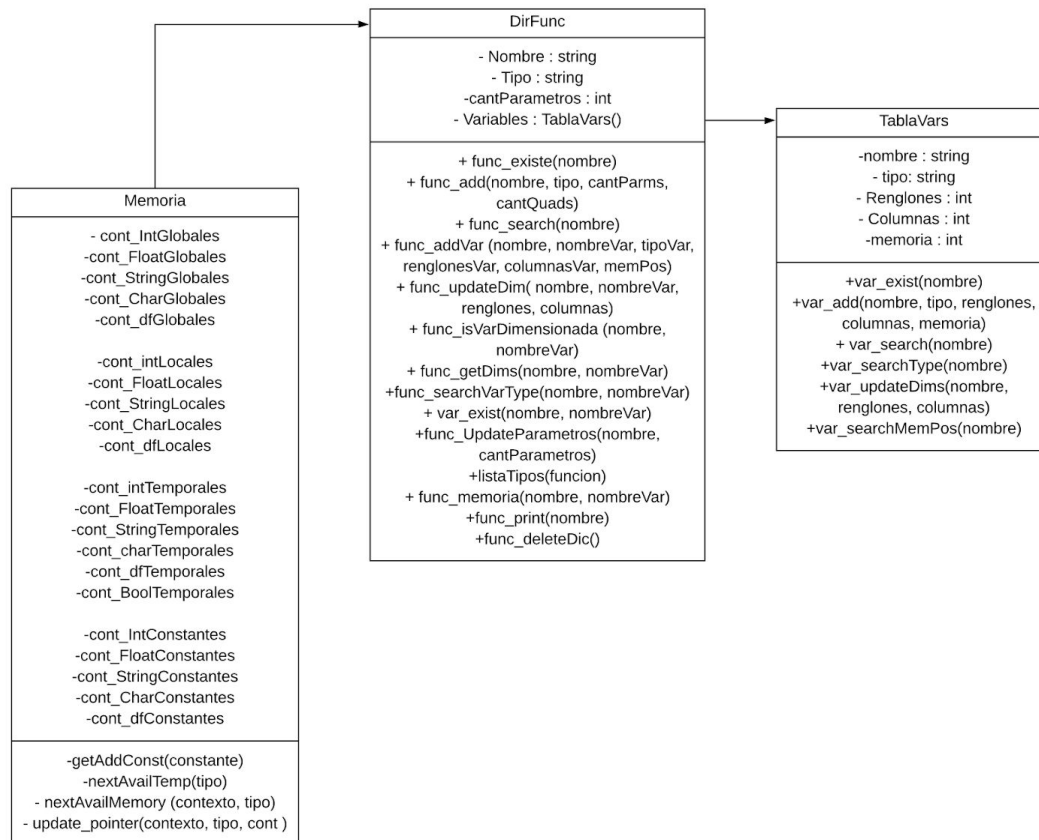
3.5.4. Tabla de consideraciones semánticas

Cubo Semántico de nuestro programa COVID, el cual representa los resultados de realizar cierto tipo de operación entre diferentes tipos de variables:

*Para efectos de simplificación, no se incluirán aquellas operaciones entre tipos que como resultado dé error.

Tipo 1	Tipo 2	Operador	Resultado
int	int	+, -, *, =	int
int	int	/	float
int	int	==, <, >, <=, >=, !=	bool
int	float	+, -, *, /	float
int	float	=	int
int	float	==, <, >, <=, >=, !=	bool
float	int	+, -, *, /, =	float
float	int	==, <, >, <=, >=, !=	bool
float	float	+, -, *, =	float
float	float	==, <, >, <=, >=, !=	bool
char	char	=	char
char	char	==, !=	bool
char, string	string, char	+	string
char, string	string, char	==, !=	bool
string	char	=	char
string	string	+, =	string
string	string	==, !=	bool
dataframe	dataframe	=	dataframe

3.6. Administración de Memoria



En la fase de compilación se crea el Directorio de Funciones, que para representarlo utilizamos la estructura de datos de diccionario. La llave representa el nombre de la función, y el valor de esa llave está compuesto por una lista de atributos semánticos: nombre, tipo, cantParametros, variables, cantQuads.

El valor del atributo de "variables" es un objeto de la TablaVars, que es la clase que representa a la Tabla de Variables. Para representar la tabla de variables, también se utilizó un diccionario, donde el valor de la llave es el nombre de la variable creada, y el valor de esa llave está compuesto por una lista de atributos semánticos: nombre, tipo, renglones, columnas y posMem. Esta última indica la posición de memoria en la que está guardada la variable, o en el caso de arreglos, indica la posición de memoria base.

La memoria de la fase de compilación es representada por contadores, que van aumentando de valor conforme se va llenando la memoria. Una vez que llega a su

límite establecido, el compilador marca error de Out of Bounds. Cada espacio de memoria de nuestro programa tiene 1000 espacios. La dividimos en secciones, comenzando con el primer bloque que representa la memoria para guardar variables globales. El segundo bloque representa la memoria para guardar variables locales. El tercer bloque es para guardar temporales generados por el compilador. En este espacio de memoria incluye guardar valores de tipo booleano. El cuarto y último bloque representa el espacio de memoria para guardar las constantes que se van encontrando durante la compilación del programa.

4. Descripción de la Máquina Virtual

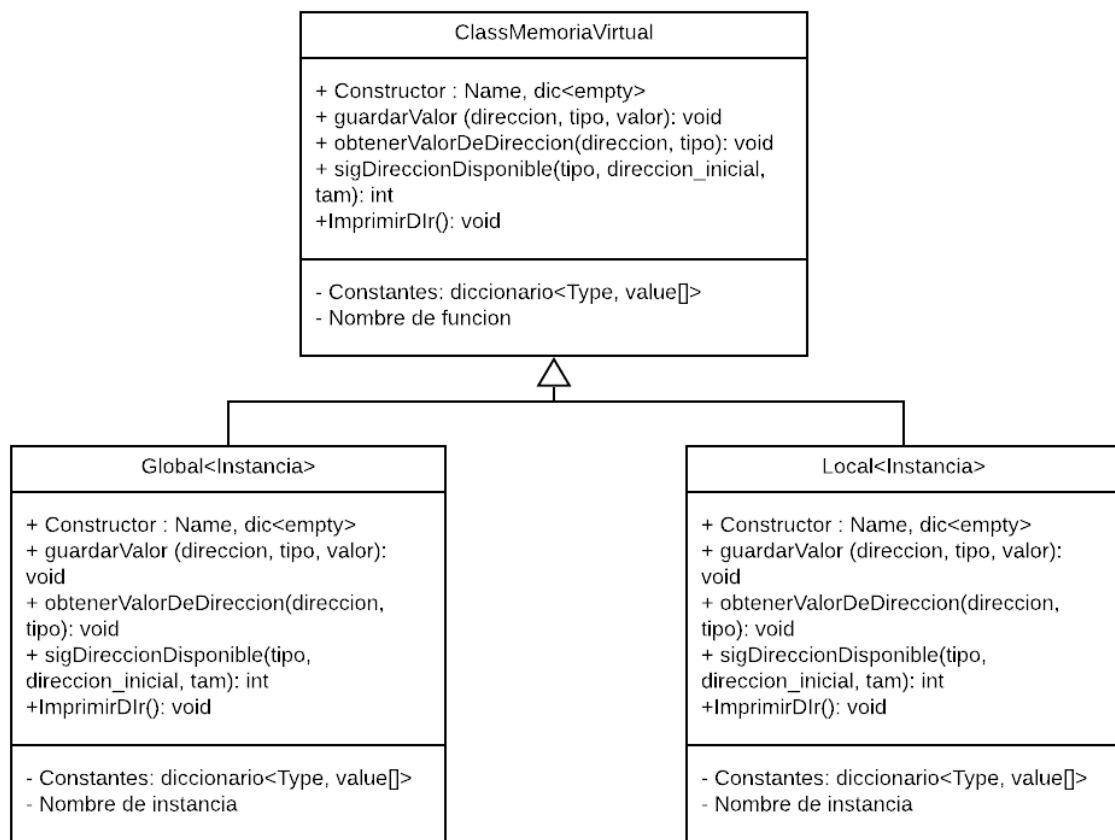
La máquina virtual se componen de la memoria virtual, que es una clase que partir del uso de diccionarios, se encarga de guardar, los valores deseados con sus respectivas direcciones y tipo de variable, el tipo de variable siendo la llave principal, donde se almacenarán todas las direcciones con sus valores, también se compone de la propia máquina virtual, la cual, con la ayuda de la memoria virtual, es la encargada de ejecutar, los cuádruplos que son recibidos, por el compilador.

4.1. Ambiente de Desarrollo

Para el desarrollo de la máquina virtual, se utilizó el lenguaje de Python, el cual da la flexibilidad suficiente, para manejar los cuádruplos, además de que se utilizaron las librerías de matplotlib, numpy, statistics, las cuales permiten hacer los diferentes cálculos

4.2. Administración de Memoria

Para la administración de memoria se creó una clase principal, la cual se encarga de guardar y administrar las direcciones y valores, en dos instancias principales, la global y la local, la global permite administrar las direcciones que se utilizan en todo el programa y la instancia local, se van declarando según haya funciones en el archivo, permitiendo administrar direcciones locales de dicha función/memoria.



4.3. Direcciones de Memoria

La direcciones de memoria que manejamos, se encuentran expresadas aquí abajo, las separamos principalmente en el tipo de sección a la que pertenece, (Global, Local, Temporal, Constantes), luego se dividió cada sección, según el tipo de variables que manejamos (int, float, string, char, data frame y en el caso de la temporal, se usa global para las condiciones).

0 - 999	int	GLOBAL
1000 - 1999	float	GLOBAL
2000 - 2999	string	GLOBAL
3000 - 3999	char	GLOBAL
4000 - 4999	dataframe	GLOBAL
5000 - 5999	int	LOCAL
6000 - 6999	float	LOCAL
7000 - 7999	string	LOCAL

8000 - 8999	char	LOCAL
9000 - 9999	dataframe	LOCAL
10000 - 10999	int	TEMPORAL
11000 - 11999	float	TEMPORAL
12000 - 12999	string	TEMPORAL
13000 - 13999	char	TEMPORAL
14000 - 14999	dataframe	TEMPORAL
15000 - 15999	bool	TEMPORAL
16000 - 16999	int	CONSTANTES
17000 - 17999	float	CONSTANTES
18000 - 18999	string	CONSTANTES
19000 - 19999	char	CONSTANTES
20000 - 20999	dataframe	CONSTANTES

5. Pruebas de Funcionamiento

Prueba: Factorial Cíclico

Código	Cuádruplos
<pre> programa factorialCic; var int : j; funcion int factorial(int n) var int : i, fact; { i = 1; fact = 1; mientras (i <= n) haz { fact = fact * i; i = i + 1; } regresa(fact); </pre>	<pre> ('GOTO', "", "", 13) ('CONS', 'int', 1, 16000) ('=', 16000, "", 5001) ('=', 16000, "", 5002) ('<=', 5001, 5000, 15000) ('GOTO', 15000, "", 11) ('*', 5002, 5001, 10000) ('=', 10000, "", 5002) ('+', 5001, 16000, 10001) ('=', 10001, "", 5001) ('GOTO', "", "", 4) ('regresa', "", "", 5002) ('ENDFUNC', "", "", "") ('CONS', 'string', "Introduce número", 18000) ('escribe', 18000, "", 'escribe') </pre>

<pre> }</pre> <pre> principal() { escribe("Introduce número"); lee(j); escribe(factorial(j)); }</pre>	<pre> ('lee', 0, "", 'lee') ('ERA', 'factorial', "", "") ('PARAMETER', 0, "", 'param1') ('GOSUB', 'factorial', 19, 1) ('=', 'factorial', "", 10000) ('escribe', 10000, "", 'escribe') ('FINPROGRAMA', "", "", "")</pre>

Prueba: Factorial Recursivo

Código	Cuádruplos
<pre> programa factorialRec; var int: j; funcion int factorial(int n) { si (n > 1) entonces { regresa(n * factorial(n-1)); } sino { regresa(1); } } principal() { escribe("Introduce número"); lee(j); escribe(factorial(j)); }</pre>	<pre> ('GOTO', "", "", 14) ('CONS', 'int', 1, 16000) ('>', 5000, 16000, 15000) ('GOTO', 15000, "", 12) ('ERA', 'factorial', "", "") ('-', 5000, 16000, 10000) ('PARAMETER', 10000, "", 'param1') ('GOSUB', 'factorial', 8, 1) ('=', 'factorial', "", 10001) ('*', 5000, 10001, 10002) ('regresa', "", "", 10002) ('GOTO', "", "", 13) ('regresa', "", "", 16000) ('ENDFUNC', "", "", "") ('CONS', 'string', "Introduce número", 18000) ('escribe', 18000, "", 'escribe') ('lee', 0, "", 'lee') ('ERA', 'factorial', "", "") ('PARAMETER', 0, "", 'param1') ('GOSUB', 'factorial', 20, 1) ('=', 'factorial', "", 10000) ('escribe', 10000, "", 'escribe') ('FINPROGRAMA', "", "", "")</pre>
<p>Resultado:</p> <p>-> "Introducenúmero"</p> <p><- 5</p> <p>-> 120</p>	

Prueba: Fibonacci Ciclico

Código	Cuádruplos
--------	------------

<pre> programa fiboCic; var int: j, w; funcion int fibonacci(int r) var int : a, b, c, contador; { contador = 0; a = 0; b = 1; mientras (contador <= r) haz { c = a + b; a = b; b = c; contador = contador + 1; } regresa (c - a); } principal() { lee(j); w = fibonacci(j); escribe(w); } </pre>	<pre> ('GOTO', "", "", 18) ('CONS', 'int', 0, 16000) ('=', 16000, "", 5004) ('=', 16000, "", 5001) ('CONS', 'int', 1, 16001) ('=', 16001, "", 5002) ('<=', 5004, 5000, 15000) ('GOTO', 15000, "", 15) ('+', 5001, 5002, 10000) ('=', 10000, "", 5003) ('=', 5002, "", 5001) ('=', 5003, "", 5002) ('+', 5004, 16001, 10001) ('=', 10001, "", 5004) ('GOTO', "", "", 6) ('-', 5003, 5001, 10002) ('regresa', "", "", 10002) ('ENDFUNC', "", "", "") ('lee', 0, "", 'lee') ('ERA', 'fibonacci', "", "") ('PARAMETER', 0, "", 'param1') ('GOSUB', 'fibonacci', 22, 1) ('=', 'fibonacci', "", 10000) ('=', 10000, "", 1) ('escribe', 1, "", 'escribe') ('FINPROGRAMA', "", "", "") </pre>
<p>Resultado:</p> <p><- 10</p> <p>-> 55</p>	

Prueba: Fibonacci Recursivo

Código	Cuádruplos
<pre> programa fiboRec; var int: j, w; funcion int fib(int n) { si (n == 1) entonces { regresa(1); } si (n == 0) entonces { </pre>	<pre> ('GOTO', "", "", 25) ('CONS', 'int', 1, 16000) ('==', 5000, 16000, 15000) ('GOTO', 15000, "", 5) ('regresa', "", "", 16000) ('CONS', 'int', 0, 16001) ('==', 5000, 16001, 15001) ('GOTO', 15001, "", 9) ('regresa', "", "", 16001) ('>', 5000, 16000, 15002) </pre>

<pre> regresa(0); } si (n > 1) entonces { regresa(fib(n-1) + fib(n-2)); } } principal() { lee(j); w = fib(j); escribe(w); } </pre>	<pre> ('GOTO', 15002, "", 24) ('ERA', 'fib', "", "") ('-', 5000, 16000, 10000) ('PARAMETER', 10000, "", 'param1') ('GOSUB', 'fib', 15, 1) ('=', 'fib', "", 10001) ('ERA', 'fib', "", "") ('CONS', 'int', 2, 16002) ('-', 5000, 16002, 10002) ('PARAMETER', 10002, "", 'param1') ('GOSUB', 'fib', 21, 1) ('=', 'fib', "", 10003) ('+', 10001, 10003, 10004) ('regresa', "", "", 10004) ('ENDFUNC', "", "", "") ('lee', 0, "", 'lee') ('ERA', 'fib', "", "") ('PARAMETER', 0, "", 'param1') ('GOSUB', 'fib', 29, 1) ('=', 'fib', "", 10000) ('=', 10000, "", 1) ('escribe', 1, "", 'escribe') ('FINPROGRAMA', "", "", "") </pre>
<p>Resultado:</p> <p><- 10</p> <p>-> 55</p>	

Prueba: Ordena y Encuentra en Arreglo

Código	Cuádruplos
<pre> programa OrdenaEncuentra; var int: j[5], h, x, n; principal() { h = 0; escribe("Tamaño del arreglo"); lee(n); x = n - 1; mientras (x > 0) haz { j[h] = 4 * 9 / x; h = h + 1; x = x - 1; } } </pre>	<pre> ('GOTO', "", "", 1) ('CONS', 'int', 0, 16000) ('=', 16000, "", 5) ('CONS', 'string', ""Tamaño del arreglo"", 18000) ('escribe', 18000, "", 'escribe') ('lee', 7, "", 'lee') ('CONS', 'int', 1, 16001) ('-', 7, 16001, 10000) ('=', 10000, "", 6) ('>', 6, 16000, 15000) ('GOTO', 15000, "", 23) ('VER', 5, 0, 4) ('+', '{0}', 5, 10001) ('CONS', 'int', 4, 16002) ('CONS', 'int', 9, 16003) </pre>

<pre> } j[2] = 23; j[1] = 33; j[4] = 8; x = n - 1; h = 0; escribe("Arreglo Desorneado: "); mientras (x > 0) haz { escribe(j[h]); h = h + 1; x = x - 1; } escribe(" - "); j[3] = 4; escribe("Encontrar el 4 : "); escribe(encontrar(j, 4)); ordena(j); escribe(" - "); escribe("Encontrar el 4: "); escribe(encontrar(j, 4)); escribe(" - "); x = n - 1; h = 0; escribe("Arreglo Ordenado: "); mientras (x > 0) haz { escribe(j[h]); h = h + 1; x = x - 1; } } </pre>	<pre> ('*', 16002, 16003, 10002) ('/', 10002, 6, 11000) ('=', 11000, "", '10001!') ('+', 5, 16001, 10003) ('=', 10003, "", 5) ('-', 6, 16001, 10004) ('=', 10004, "", 6) ('GOTO', "", "", 9) ('CONS', 'int', 2, 16004) ('VER', 16004, 0, 4) ('+', '{0}', 16004, 10005) ('CONS', 'int', 23, 16005) ('=', 16005, "", '10005!') ('VER', 16001, 0, 4) ('+', '{0}', 16001, 10006) ('CONS', 'int', 33, 16006) ('=', 16006, "", '10006!') ('VER', 16002, 0, 4) ('+', '{0}', 16002, 10007) ('CONS', 'int', 8, 16007) ('=', 16007, "", '10007!') ('-', 7, 16001, 10008) ('=', 10008, "", 6) ('=', 16000, "", 5) ('CONS', 'string', '"Arreglo Desorneado: "', 18001) ('escribe', 18001, "", 'escribe') ('>', 6, 16000, 15001) ('GOTO', 15001, "", 51) ('VER', 5, 0, 4) ('+', '{0}', 5, 10009) ('escribe', '10009!', "", 'escribe') ('+', 5, 16001, 10010) ('=', 10010, "", 5) ('-', 6, 16001, 10011) ('=', 10011, "", 6) ('GOTO', "", "", 41) ('CONS', 'string', '" - "', 18002) ('escribe', 18002, "", 'escribe') ('CONS', 'int', 3, 16008) ('VER', 16008, 0, 4) ('+', '{0}', 16008, 10012) ('=', 16002, "", '10012!') ('CONS', 'string', '"Encontrar el 4 : "', 18003) ('escribe', 18003, "", 'escribe') ('encontrar', 0, '%5#4', 10013) ('escribe', 10013, "", 'escribe') ('ordena', 0, 5, 10014) </pre>
--	--

	('escribe', 18002, "", 'escribe') ('CONS', 'string', '"Encontrar el 4: "', 18004) ('escribe', 18004, "", 'escribe') ('encontrar', 0, '%5#4', 10015) ('escribe', 10015, "", 'escribe') ('escribe', 18002, "", 'escribe') ('-', 7, 16001, 10016) ('=', 10016, "", 6) ('=', 16000, "", 5) ('CONS', 'string', '"Arreglo Ordenado: "', 18005) ('escribe', 18005, "", 'escribe') ('>', 6, 16000, 15002) ('GOTO', 15002, "", 83) ('VER', 5, 0, 4) ('+', '{0}', 5, 10017) ('escribe', '10017!', "", 'escribe') ('+', 5, 16001, 10018) ('=', 10018, "", 5) ('-', 6, 16001, 10019) ('=', 10019, "", 6) ('GOTO', "", "", 73) ('FINPROGRAMA', "", "", "")
Resultado: -> "Tamañodelarreglo" <- 5 -> "ArregloDesorneado:" -> 9.0 -> 33 -> 23 -> 36.0 -> "-" -> "Encontrarel4:" -> 3 -> "-" -> "Encontrarel4:" -> 0 -> "-" -> "ArregloOrdenado:" -> 4 -> 9 -> 23 -> 33	

Prueba: Funciones Especiales Estadísticas

Código	Cuádruplos
--------	------------

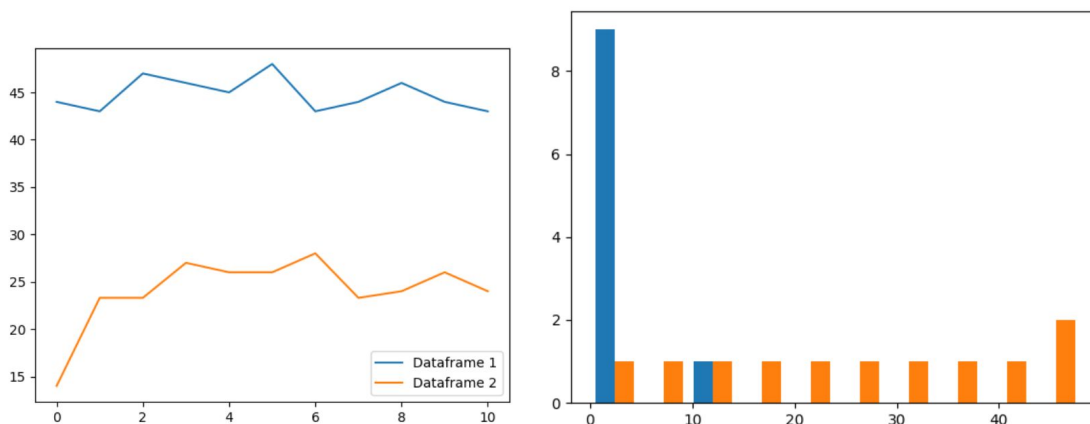
<pre> programa FunEsp; var dataframe : i, j; float: h, f, c; principal() { cargaArchivo(i, "pruebas/dataframes/dt1.txt", 1, 5); cargaArchivo(j, "pruebas/dataframes/dt2.txt", 1, 3); h = mediana(i,-1,-1); f = mediana(j,-1,-1); escribe(h); escribe(f); h = moda(i,-1,-1); f = moda(j,-1,-1); escribe(h); escribe(f); h = varianza(i,-1,-1); f = varianza(j,-1,-1); escribe(h); escribe(f); h = media(i,-1,-1); f = media(j,-1,-1); escribe(h); escribe(f); c = correlacion(i, j, -1,-1); escribe(c); } </pre>	<pre> ('CONS', 'dataframe', 'i', 20000) ('CONS', 'dataframe', 'j', 20001) ('GOTO', "", "", 3) ('CONS', 'string', "'pruebas/dataframes/dt1.txt'", 18000) ('CONS', 'int', 1, 16000) ('CONS', 'int', 5, 16001) ('carga', 20000, "'pruebas/dataframes/dt1.txt'", "") ('CONS', 'string', "'pruebas/dataframes/dt2.txt'", 18001) ('CONS', 'int', 3, 16002) ('carga', 20001, "'pruebas/dataframes/dt2.txt'", "") ('mediana', 20000, '%0#0', 11000) ('=', 11000, "", 1000) ('mediana', 20001, '%0#0', 11001) ('=', 11001, "", 1001) ('escribe', 1000, "", 'escribe') ('escribe', 1001, "", 'escribe') ('moda', 20000, '%0#0', 11002) ('=', 11002, "", 1000) ('moda', 20001, '%0#0', 11003) ('=', 11003, "", 1001) ('escribe', 1000, "", 'escribe') ('escribe', 1001, "", 'escribe') ('varianza', 20000, '%0#0', 11004) ('=', 11004, "", 1000) ('varianza', 20001, '%0#0', 11005) ('=', 11005, "", 1001) ('escribe', 1000, "", 'escribe') ('escribe', 1001, "", 'escribe') ('media', 20000, '%0#0', 11006) ('=', 11006, "", 1000) ('media', 20001, '%0#0', 11007) ('=', 11007, "", 1001) ('escribe', 1000, "", 'escribe') ('escribe', 1001, "", 'escribe') ('correlacion', '%20000#20001', '%0#0', 11008) ('=', 11008, "", 1002) ('escribe', 1002, "", 'escribe') ('FINPROGRAMA', "", "", "") </pre>
<p>Resultado:</p> <pre> -> 24.5 -> 44.5 -> 23.3 -> 44 -> 10.103958333333333 -> 117.9625 </pre>	

-> 24.34375
-> 42.3125
-> -0.2031814810400866

Prueba: Funciones Graficas

Código	Cuádruplos
<pre> programa graphFuncs; var dataframe: j, i; principal() { cargaArchivo(i, "pruebas/dataframes/dt1.txt", 1, 5); cargaArchivo(j, "pruebas/dataframes/dt2.txt", 1, 3); plotline(j, i, 10); histograma(j, -1, -1); } </pre>	<pre> ('CONS', 'dataframe', 'j', 20000) ('CONS', 'dataframe', 'i', 20001) ('GOTO', "", "", 3) ('CONS', 'string', '"pruebas/dataframes/dt1.txt"', 18000) ('CONS', 'int', 1, 16000) ('CONS', 'int', 5, 16001) ('carga', 20001, '"pruebas/dataframes/dt1.txt"', "") ('CONS', 'string', '"pruebas/dataframes/dt2.txt"', 18001) ('CONS', 'int', 3, 16002) ('carga', 20000, '"pruebas/dataframes/dt2.txt"', "") ('histograma', 20000, 0, 0) ('FINPROGRAMA', "", "", "") </pre>

Resultado:



6. Documentación

6.1. Compilación

Funciones más representativas de la fase de Compilación:

```
'''
Generacion de cuadрупlos
'''
def QuadGenerate(operator, leftOperand, rightOperand, result):
    QuadTemporal = (operator, leftOperand, rightOperand, result)
    pushQuad(QuadTemporal)
    NumQuad = nextQuad() - 1
    print(">> Quad {}: ('{}','{}','{}','{}')".format(NumQuad, operator, leftOperand,
rightOperand, result))

    print("\n")
```

```
'''
Regresa el siguiente temporal disponible, dependiendo el tipo
'''
def nextAvailTemp(tipo):
    global cont_IntTemporales
    global cont_FloatTemporales
    global cont_BoolTemporales
    global avail

    if tipo == 'int':
        if cont_IntTemporales < limite_intTemporales: #Mientras tenga espacio en
la memoria, podra seguir pidiendo temporales
            avail = cont_IntTemporales
            cont_IntTemporales += 1
        else:
            errorOutOfBounds('temporales','Enterass')
    elif tipo == 'float':

        if cont_FloatTemporales < limite_floatTemporales:
            avail = cont_FloatTemporales
            cont_FloatTemporales += 1
        else:
            errorOutOfBounds('temporales','Flotantes')

    elif tipo == 'bool':
        if cont_BoolTemporales < limite_boolTemporales:
```



```

        avail = cont_BoolTemporales
        cont_BoolTemporales = cont_BoolTemporales + 1
    else:
        errorOutOfBounds('temporales', 'Boleanas')
    else:
        avail = -1
        print("Error: Tipo de variable no existente")
    return avail

```

```

'''
Regresa el siguiente espacio de memoria disponible
'''
def nextAvailMemory(contexto, tipo):
    global cont_IntGlobales
    global cont_IntLocales
    global cont_FloatGlobales
    global cont_FloatLocales
    global cont_StringGlobales
    global cont_StringLocales
    global cont_CharGlobales
    global cont_CharLocales
    global cont_dfConstantes
    global cont_dfLocales

    posMem = -1

    #Global
    if contexto == GBL:

        if tipo == 'int':
            if cont_IntGlobales < limite_intGlobales:
                posMem = cont_IntGlobales
                cont_IntGlobales += 1
            else:
                errorOutOfBounds(GBL, 'Enterass')

        elif tipo == 'float':
            if cont_FloatGlobales < limite_floatGlobales:

```

```

        posMem = cont_FloatGlobales
        cont_FloatGlobales += 1
    else:
        errorOutOfBounds(GBL, 'Floats')

elif tipo == 'string':
    if cont_StringGlobales < limite_stringsGlobales:
        posMem = cont_StringGlobales
        cont_StringGlobales += 1
    else:
        errorOutOfBounds(GBL, 'Strings')

elif tipo == 'char':
    if cont_CharGlobales < limite_charGlobales:
        posMem = cont_CharGlobales
        cont_CharGlobales += 1
    else:
        errorOutOfBounds(GBL, 'Chars')

elif tipo == 'dataframe':
    if cont_dfConstantes < limite_dfConstantes:
        posMem = cont_dfConstantes
        cont_dfConstantes += 1
    else:
        errorOutOfBounds(GBL, 'Dataframes')
#Locales
else:
    if tipo == 'int':
        if cont_IntLocales < limite_intLocales:
            posMem = cont_IntLocales
            cont_IntLocales += 1
        else:
            errorOutOfBounds('Locales', 'Enteradas')

    elif tipo == 'float':
        if cont_FloatLocales < limite_floatLocales:
            posMem = cont_FloatLocales
            cont_FloatLocales += 1
        else:
            errorOutOfBounds('Locales', 'Floats')

```

```

elif tipo == 'string':
    if cont_StringLocales < limite_stringsLocales:
        posMem = cont_StringLocales
        cont_StringLocales += 1
    else:
        errorOutOfBounds('Locales', 'Strings')

elif tipo == 'char':
    if cont_CharLocales < limite_charLocales:
        posMem = cont_CharLocales
        cont_CharLocales += 1
    else:
        errorOutOfBounds('Locales', 'Chars')

elif tipo == 'dataframe':
    if cont_dfConstantes < limite_dfConstantes:
        posMem = cont_dfConstantes
        cont_dfConstantes += 1
    else:
        errorOutOfBounds('Locales', 'Dataframes')

return posMem

```

```

'''
Modifica la posicion del pointer de acuerdo al contexto, tipo y contador dado
'''
def update_pointer(contexto, tipo, cont):
    global cont_IntGlobales
    global cont_IntLocales
    global cont_FloatGlobales
    global cont_FloatLocales
    global cont_StringGlobales
    global cont_StringLocales
    global cont_CharGlobales
    global cont_CharLocales
    global cont_dfConstantes

    if contexto == GBL:

```

```

if tipo == 'int':
    cont_IntGlobales += cont
    if cont_IntGlobales > limite_intGlobales:
        sys.exit('Error: Overflow Enteras Globales')

if tipo == 'float':
    cont_FloatGlobales += cont
    if cont_FloatGlobales > limite_floatGlobales:
        sys.exit('Error: Overflow Flotantes Globales')

if tipo == 'string':
    cont_StringGlobales += cont
    if cont_StringGlobales > limite_stringsGlobales:
        sys.exit('Error: Overflow Strings Globales')

if tipo == 'char':
    cont_CharGlobales += cont
    if cont_CharGlobales > limite_charGlobales:
        sys.exit('Error: Overflow Chars Globales')

if tipo == 'dataframe':
    cont_dfConstantes += cont
    if cont_dfConstantes > limite_dfConstantes:
        sys.exit('Error: Overflow DF Globales')
else:
    if tipo == 'int':
        cont_IntLocales += cont
        if cont_IntLocales > limite_intLocales:
            sys.exit('Error: Overflow Enteras Locales')

    if tipo == 'float':
        cont_FloatLocales += cont
        if cont_FloatLocales > limite_floatLocales:
            sys.exit('Error: Overflow Flotantes Locales')

    if tipo == 'string':
        cont_StringLocales += cont
        if cont_StringLocales > limite_stringsLocales:
            sys.exit('Error: Overflow Strings Locales')

```

```

        if tipo == 'char':
            cont_CharLocales += cont
            if cont_CharLocales > limite_charLocales:
                sys.exit('Error: Overflow Chars Locales')

        if tipo == 'dataframe':
            cont_dfConstantes += cont
            if cont_dfConstantes > limite_dfConstantes:
                sys.exit('Error: Overflow DF Locales')

```

```

'''
Agrega la nueva variable a la tabla de variables
'''
def p_pn_2_addVariable(p):
    '''
    pn_2_addVariable :
    '''
    global currentFunc
    global varName
    global currentType
    global currentVarName
    global currentCantVars
    global boolDataf

    varName = p[-1] #Toma el nombre de la variable
    currentVarName = varName #Asigna a la variable currentVarName el valor de la
variable actual

    PosMem = nextAvailMemory(currentFunc, currentType) #Pide el siguiente espacio
de memoria disponible para almacenar la variable, de acuerdo a su tipo

    directorioFunciones.func_addVar(currentFunc, varName, currentType, 0, 0,
PosMem) #Ejecuta la funcion func_addVar del directorio de funciones para agregar
la nueva variable a la tabla de variables

    currentCantVars += 1

    if boolDataf: #Si la variable es un dataframe, genera un cuadruplo para
identificarlo

```

```
QuadGenerate('CONS', 'dataframe', varName, PosMem)
```

```
'''
Agega nueva funcion al Directorio de Funciones
'''
def p_pnFunDecl(p):
    '''
    pnFunDecl :
    '''
    global currentFunc
    global currentType
    global currentCantParams
    global currentCantVars
    global returnBool

    currentCantVars = 0
    currentCantParams = 0
    currentFunc = p[-1] #Toma el nombre de la funcion
    directorioFunciones.func_add(currentFunc, currentType, currentCantParams,
nextQuad()) #Ejecuta la funcion del directorio de funciones func_add para agregar
la funcion leida al diccionario de Directorio de Funciones

    if directorioFunciones.directorio_funciones[currentFunc]['tipo'] == 'void':
#Si la función es Void, desactiva el booleano returnBool para indicar que no
tiene que esperar un Retorno. De lo contrario, lo activa.
        returnBool = False
    else:
        returnBool = True
```

```
'''Checa si el top del POper es un + o - para generar el cuadruplo de esa
operacion'''
def p_pnExp4(p):
    '''
    pnExp4 :
    '''
    if topOperador() in OP_SUMARESTA:
        quad_rightOperand = popOperandos()
        quad_rightType = popTipos()
```

```

quad_rightMem = popMemoria()
quad_leftOperand = popOperandos()
quad_leftMem = popMemoria()
quad_leftType = popTipos()
quad_operator = popOperadores()

global cuboSem
quad_resultType = cuboSem.getType(quad_leftType, quad_rightType,
quad_operator) #Verifica que la operacion sea valida

if quad_resultType == 'error':
    errorTypeMismatch()
else: #Si la operacion es valida, genera el cuadruplo
    temporal = nextAvailTemp(quad_resultType)
    QuadGenerate(quad_operator, quad_leftMem, quad_rightMem, temporal)
    pushOperando(temporal)
    pushMemoria(temporal)
    pushTipo(quad_resultType)

```

```

class DirFunc:
    '''
    Constructor que inicializa el diccionario directorio_funciones con sus
    atributos, empezando con la seccion de globales.
    '''
    def __init__(self):
        self.directorio_funciones = {'global': {'nombre' : 'global', 'tipo' :
'void', 'cantParametros' : 0, 'variables': TablaVars(), 'cantQuads' : 0}}
        print("Funcion creada exitosamente: Global de tipo void")

    '''
    Funcion que checa si ya existe una funcion en el directorio_funciones
    '''
    def func_existe(self, nombre):
        return nombre in self.directorio_funciones.keys()

    '''
    Funcion para agregar una funcion nueva al diccionario de directorio_funciones
    '''

```

```

def func_add(self, nombre, tipo, cantParametros, cantQuads):
    if self.func_existe(nombre):
        print ("Error: Declaracion multiple de funcion: ", str(nombre), "\n")
    else:
        self.directorio_funciones[nombre] = {
            'nombre': nombre,
            'tipo': tipo,
            'cantParametros': cantParametros,
            'variables': TablaVars(),
            'cantQuads': cantQuads
        }
        print ("NUEVA Funcion creada en el Directorio de Funciones: ", nombre,
" de tipo: ", tipo, "\n")

'''
    Funcion que regresa todos los datos de la funcion dada. Si no existe no
regresa nada.
'''

def func_search(self, nombre):
    if self.func_existe(nombre):
        return self.directorio_funciones[nombre]
    else:
        return None

'''
    Funcion que agrega una variable a la tabla de variables linkeada a la funcion
dada.
'''

def func_addVar(self, nombre, nombreVar, tipoVar, renglonesVar, columnasVar,
memPos):

    if self.directorio_funciones[nombre]['variables'].var_add(nombreVar,
tipoVar, renglonesVar, columnasVar, memPos):
        print ("Variable: ", nombreVar, " creada exitosamente, dentro de la
funcion: ", nombre)
    else:
        print ("Error: No es posible crear la variable: ", nombreVar, " dentro
de la funcion: ", nombre)

'''

```



```

Funcion para actualizar indicar que una variable es dimensionada
'''

def func_updateDim(self, nombre, nombreVar, renglones, columnas):
    if self.directorio_funciones[nombre]['variables'].var_exist(nombreVar):
        if columnas > 0:
            return
self.directorio_funciones[nombre]['variables'].var_updateDims(nombreVar,
renglones, columnas)
        else:
            return
self.directorio_funciones[nombre]['variables'].var_updateDims(nombreVar,
renglones, -1)
        else:
            print("Error: Variable ", nombreVar, "no existe en este contexto ",
nombre)
            return None

'''

Funcion que dice si una variable dada es dimensionada en el contexto dado
'''

def func_isVarDimensionada(self, nombre, nombreVar):
    if self.directorio_funciones[nombre]['variables'].var_exist(nombreVar):
        print("DIRECTORIO FUNC DE ", nombreVar, "col: ",
self.directorio_funciones[nombre]['variables'].tabla_variables[nombreVar]['column
as'])

        if
self.directorio_funciones[nombre]['variables'].tabla_variables[nombreVar]['column
as'] > 0 or
self.directorio_funciones[nombre]['variables'].tabla_variables[nombreVar]['renglo
nes'] > 0:
            return 1
        else:
            return 0
    else:
        return -1 #No existe esa variable en este contexto

'''

Funcion que regresa las dimensiones de una variable dimensionada en forma de

```

```

lista
'''
def func_getDims(self, nombre, nombreVar):
    if self.directorio_funciones[nombre]['variables'].var_exist(nombreVar):
        dim =
[self.directorio_funciones[nombre]['variables'].tabla_variables[nombreVar]['columnas'],
self.directorio_funciones[nombre]['variables'].tabla_variables[nombreVar]['renglones']]

        return dim
    else:
        return -1 #No existe esa variable en este contexto

'''
Funcion que regresa el string del tipo de una variable previamente creada en
las funciones
'''
def func_searchVarType(self, nombre, nombreVar):
    if self.directorio_funciones[nombre]['variables'].var_exist(nombreVar):
        return
self.directorio_funciones[nombre]['variables'].var_searchType(nombreVar)
    else:
        print("Error: Variable: ", nombreVar ," no existe en este contexto: ",
nombre)

        return None

'''
Funcion que regresa si existe una variable en la tabla de variables de la
funcion dada
'''
def var_exist(self, nombre, nombreVar):
    if self.directorio_funciones[nombre]['variables'].var_exist(nombreVar):
        return True
    else:
        return False

'''
Funcion para cambiar la cantidad de parametros de una funcion dada.
'''
def func_UpdateParametros(self, nombre, cantParametros):

```

```

        if self.func_existe(nombre):
            self.directorio_funciones[nombre]['cantParametros'] = cantParametros
        else:
            print("Error: NO existe la funcion: ", nombre)

'''

'''

def listaTipos(self, funcion):
    return
[self.directorio_funciones[funcion]['variables'].tabla_variables[x]['tipo'] for x
in self.directorio_funciones[funcion]['variables'].tabla_variables]

'''

Funcion que regresa la posicion de memoria virtual de la variable dada
'''

def func_memoria(self, nombre, nombreVar):
    if self.directorio_funciones[nombre]['variables'].var_exist(nombreVar):
        return
self.directorio_funciones[nombre]['variables'].var_searchMemPos(nombreVar)
    else:
        print("Error: Variable: ", nombreVar, "no existe en este contexto: ",
nombre)

'''

Funcion que imprime el directorio de funciones actual
'''

def func_print(self, nombre):
    print (self.directorio_funciones[nombre]['variables'].tabla_variables)
    print("\n")

'''

Funcion que borra el directorio de funciones
'''

def func_deleteDic(self):
    self.directorio_funciones.clear()

```

```

class TablaVars:

    """
    Constructor para inicializar el diccionario tabla_variables
    """

    def __init__(self):
        self.tabla_variables = {}

    """
    Funcion que verifica si una variable ya existe
    """
    def var_exist(self, nombre):
        return nombre in self.tabla_variables.keys()

    """
    Funcion para agregar una nueva variable a la tabla
    """
    def var_add(self, nombre, tipo, renglones, columnas, memoria ):
        if self.var_exist(nombre):
            print("Error : Variable ", str(nombre), " duplicada")
            return False
        else:
            self.tabla_variables[nombre] = {
                'nombre' : nombre,
                'tipo': tipo,
                'renglones': renglones,
                'columnas' : columnas,
                'memoria' : memoria
            }
            return True

    """
    Funcion que regresa todos los datos de una variable (si existe)
    """
    def var_search(self, nombre):
        if self.var_exist(nombre):
            return self.tabla_variables[nombre]
        else:
            return None

```

```

'''
Funcion que regresa el el valor del atributo "Tipo" de la variable dada
'''
def var_searchType(self, nombre):
    if self.var_exist(nombre):
        return self.tabla_variables[nombre]['tipo']
    else:
        return None

'''
Funcion para indicar que una variable es dimensionada cambiando los valores de
los atributos de renglones y columnas
'''
def var_updateDims(self, nombre, renglones, columnas):

    if self.var_exist(nombre):

        if columnas < 0:
            self.tabla_variables[nombre]['renglones'] = renglones
        else:
            self.tabla_variables[nombre]['columnas'] = columnas

        print("Dimensiones actualizadas de la variable: ", nombre)
        print("Renglones: ", self.tabla_variables[nombre]['renglones'],
"Columnas: ", self.tabla_variables[nombre]['columnas'])

    else:
        print("Error. No es posible actualizar las dimensaiones de una
variable que no existe: ", nombre)

'''
Funcion que regresa la posicion de memoria virtual de la variable dada
'''
def var_searchMemPos(self, nombre):
    if self.var_exist(nombre):
        return self.tabla_variables[nombre]['memoria']
    else:
        return None

```

6.2. Máquina Virtual

Función principal de la máquina virtual, encargada de detectar el parametro del cuádruplo y correrlo.

```
def correr():
    # Se declaran las variables globales a utilizar
    global constLista
    global cuaLista
    global cuaIndice
    global cuádruplo
    global mem_GLOBAL
    global pilaRetorno
    global sigCuaIndice
    global pilaCorriendo

    # Ciclo que permite guardar todos los constantes antes de correr
    los demas cuádruplo
    for cons in constLista:
        llenarValor(mem_GLOBAL, cons[3], cons[1], cons[2])

    terminado = False # nos avisas cuando salir del programa
    while not terminado:
        sigCuaIndice = -1 # nos permite llevar control, de que cuadro
        ejecutar
        pilaCorriendo = top(CONST_EJECUCION) # Saca la instancia de
        memoria que se este ejecutando
        cuádruplo = cuaLista[cuaIndice] # Saca el cuádruplo a ejecutar

        # ASIGNACION
        if cuádruplo[0] == '=':
            try: # Sino encuentra el valor, checa que este en la pila
                de valores de retorno
                valor = getValor(pilaCorriendo, cuádruplo[1],
                                getTipo(cuádruplo[1]))
            except:
                valor = pop(CONST_RETORNO_VALOR)

            llenarValor(pilaCorriendo, cuádruplo[3],
                        getTipo(cuádruplo[3]), valor)
```

```

# COMANDOS
# CONS
# Se saltará porque ya se agregaron con antelación
# GOTO
elif cuadruplo[0] == 'GOTO':
    sigCuaIndice = int(cuadruplo[3])
# GOTOF
elif cuadruplo[0] == 'GOTOF':
    auxValor = getValor(pilaCorriendo, cuadruplo[1],
getTipo(cuadruplo[1]))
    if not auxValor: #Confirma si es falso el valor
        sigCuaIndice = int(cuadruplo[3])
# GOSUB
elif cuadruplo[0] == 'GOSUB':
    pilaCorriendo = pop(CONST_TEMPORAL) # Sacamos la instancia
de memoria temporal
    push(CONST_EJECUCION, pilaCorriendo) # Metemos la instancia
a ejecucion
    push(CONST_FUNCION_RETORNO, cuadruplo[2]) # Guardamos el
retorno
    sigCuaIndice = int(cuadruplo[3])
# ERA
elif cuadruplo[0] == 'ERA':
    #Declara nueva funcion de memoria virtual
    memNueva = memVirtual(str(cuadruplo[1]))
    push(CONST_TEMPORAL, memNueva)
# PARAMETER
elif cuadruplo[0] == 'PARAMETER':
    tipo = getTipo(cuadruplo[1])
    valor = getValor(pilaCorriendo, cuadruplo[1], tipo)
    auxMem = top(CONST_TEMPORAL)
    # Nos permite saber que direccion sigue
    # Donde empiezan las variables globales
    direccion = auxMem.sigDireccionDisponible(tipo, 5000,
ESPACIO_MEMORIA)
    llenarValor(auxMem, direccion, getTipo(direccion), valor)
# ENDFUNC
elif cuadruplo[0] == 'ENDFUNC':
    pop(CONST_EJECUCION)
    sigCuaIndice = int(pop(CONST_FUNCION_RETORNO))

```

```

        # regresa
        elif cuadruplo[0] == 'regresa':
            valor = getValor(pilaCorriendo, cuadruplo[3],
getTipo(cuadruplo[3]))
            push(CONST_RETORNO_VALOR, str(valor)) #Guarda el valor para
despues
            pop(CONST_EJECUCION) #Sacar funcion de la pila, porque se
termino de ejecutar
            sigCuaIndice = int(pop(CONST_FUNCION_RETORNO));
        # lee
        elif cuadruplo[0] == 'lee':
            texto = input("<- ")
            #Verifica que tipo de valor es el que recibe, para
guardarlo donde corresponde
            try:
                int(texto)
                tipo = 'int'
            except:
                try:
                    float(texto)
                    tipo = 'float'
                except:
                    try:
                        str(texto)
                        tipo = 'char' if len(texto) == 1 else 'string'
                    except:
                        print("Error Maquina Virtual:
{}".format(sys.exc_info()[0], cuaIndice))
                auxTipo = getTipo(cuadruplo[1])
                if tipo == auxTipo:
                    llenarValor(pilaCorriendo, cuadruplo[1], auxTipo,
texto)
                else:
                    print("Error Maquina Virtual: {} es diferente a
{}".format(tipo, auxTipo))
                    sys.exit()
                return
        # escribe
        elif cuadruplo[0] == 'escribe':
            #Trae el valor y lo imprime

```



```

        texto = getValor(pilaCorriendo, cuadruplo[1],
getTipo(cuadruplo[1]))
        print("->",str(texto))

        #ARREGLO
        #VER
        elif cuadruplo[0] == 'VER':
            valor = int(getValor(pilaCorriendo, cuadruplo[1],
getTipo(cuadruplo[1])))
            if valor != 0:
                valor = valor - 1 # Para que se vuelva índice y se
prueba.

            if valor > int(cuadruplo[3]) or valor < int(cuadruplo[2]):
                print("Error Maquina Virtual: El valor {} no pertenece a
los indices.".format(valor+1))
                sys.exit()
                return
        elif cuadruplo[0] == 'ordena':
            inicio = int(cuadruplo[1])
            tam = int(cuadruplo[2])
            tipo = getTipo(cuadruplo[1])
            arr = []
            #Crea arreglo a partir de los datos de memoria
            for i in range (0, tam-1):
                valor = getValor(pilaCorriendo, inicio + i, tipo)
                # Verifca que valor utilizar
                try:
                    int(valor)
                    valor = int(valor)
                except:
                    try:
                        float(valor)
                        valor = float(valor)
                    except:
                        try:
                            str(valor)
                            tipo = 'char' if len(valor) == 1 else
'string'
                        except:
                            print("Error Maquina Virtual:
{}".format(sys.exc_info()[0], cuaIndice))

```

```

        arr.append(valor)
    arr.sort()
    for i in range (0, tam-1): llenarValor(pilaCorriendo,
inicio + i, tipo, arr[i])
    elif cuadruplo[0] == 'encontrar':
        inicio = int(cuadruplo[1])
        tipo = getTipo(cuadruplo[1])
        aux = cuadruplo[2].split('#')
        tam = int(aux[0][1:])
        buscado = int(aux[1][:])
        arr = []
        #Crea arreglo a partir de los datos de memoria
        for i in range (0, tam-1):
            valor = getValor(pilaCorriendo, inicio + i, tipo)
            # Verifca que valor utilizar
            try:
                int(valor)
                valor = int(valor)
            except:
                try:
                    float(valor)
                    valor = float(valor)
                except:
                    try:
                        str(valor)
                        valor = 'char' if len(valor) == 1 else
'string'
                    except:
                        print("Error Maquina Virtual:
{}".format(sys.exc_info()[0], cuaIndice))
            arr.append(valor)
            llenarValor(pilaCorriendo, cuadruplo[3], tipo,
arr.index(buscado))

    # FUNCIONES ESPECIALES
    # Mediana
    elif cuadruplo[0] == 'mediana':
        subArreglo = []
        tipo = getTipo(cuadruplo[3])
        arreglo = getValor(pilaCorriendo, cuadruplo[1],
getTipo(cuadruplo[1]))

```

```

        aux = cuadruplo[2].split('#')
        if (aux[0] != '0' and aux[1] != '0'): #Si vienen 0
significa que es todo el arreglo
            de = int(aux[0][1:])
            a = int(aux[1][:])
            # Si es -1, significa que es 0
            if (aux[0] == -1):
                de = 0
            if (aux[1] == -1):
                a = 0

            if verificar(arreglo, de, a): # Verificar con cualquier
arreglo, ya que deben de ser del mismo tamaño
                for r in range(de,a+1):
subArreglo.append(arreglo[r]) #Crea arreglo a partir de los datos
sacados

                try:
                    llenarValor(pilaCorriendo, cuadruplo[3], tipo,
statistics.median(subArreglo))
                except:
                    print("Error Maquina Virtual: El arreglo no
tiene mediana, en el rango de {} a {}".format(de, a))
                    sys.exit()
                    return

            else:
                try:
                    llenarValor(pilaCorriendo, cuadruplo[3], tipo,
statistics.median(arreglo))
                except:
                    print("Error Maquina Virtual: El arreglo no tiene
mediana en todo el archivo")
                    sys.exit()
                    return

        # Media
        elif cuadruplo[0] == 'media':
            subArreglo = []
            tipo = getTipo(cuadruplo[3])
            arreglo = getValor(pilaCorriendo, cuadruplo[1],
getTipo(cuadruplo[1]))
            aux = cuadruplo[2].split('#')

```

```

        if (aux[0] != '0' and aux[1] != '0'): #Si vienen 0
significa que es todo el arreglo
            de = int(aux[0][1:])
            a = int(aux[1][:])
            # Si es -1, significa que es 0
            if (aux[0] == -1):
                de = 0
            if (aux[1] == -1):
                a = 0
            if verificar(arreglo, de, a): # Verificar con cualquier
arreglo, ya que deben de ser del mismo tamaño
                for r in range(de,a+1):
subArreglo.append(arreglo[r]) #Crea arreglo a partir de los datos
sacados

                try:
                    llenarValor(pilaCorriendo, cuadruplo[3], tipo,
statistics.mean(subArreglo))
                except:
                    print("Error Maquina Virtual: El arreglo no
tiene mediana, en el rango de {} a {}".format(de, a))
                    sys.exit()
                    return

            else:
                try:
                    llenarValor(pilaCorriendo, cuadruplo[3], tipo,
statistics.mean(arreglo))
                except:
                    print("Error Maquina Virtual: El arreglo no tiene
mediana en todo el archivo")
                    sys.exit()
                    return

# moda
elif cuadruplo[0] == 'moda':
    subArreglo = []
    tipo = getTipo(cuadruplo[3])
    arreglo = getValor(pilaCorriendo, cuadruplo[1],
getTipo(cuadruplo[1]))
    aux = cuadruplo[2].split('#')
    if (aux[0] != '0' and aux[1] != '0'): #Si vienen 0
significa que es todo el arreglo

```

```

        de = int(aux[0][1:])
        a = int(aux[1][:])
        # Si es -1, significa que es 0
        if (aux[0] == -1):
            de = 0
        if (aux[1] == -1):
            a = 0
        if verificar(arreglo, de, a): # Verificar con cualquier
arreglo, ya que deben de ser del mismo tamaño
            for r in range(de,a+1):
subArreglo.append(arreglo[r]) #Crea arreglo a partir de los datos
sacados

            try:
                llenarValor(pilaCorriendo, cuadruplo[3], tipo,
statistics.mode(subArreglo))
            except:
                print("Error Maquina Virtual: El arreglo no
tiene moda o tiene mas de una, en el rango de {} a {}".format(de, a))
                sys.exit()
                return

        else:
            try:
                llenarValor(pilaCorriendo, cuadruplo[3], tipo,
statistics.mode(arreglo))
            except:
                print("Error Maquina Virtual: El arreglo no tiene
moda o tiene mas de una, en todo el archivo")
                sys.exit()
                return

    # varianza
    elif cuadruplo[0] == 'varianza':
        subArreglo = []
        tipo = getTipo(cuadruplo[3])
        arreglo = getValor(pilaCorriendo, cuadruplo[1],
getTipo(cuadruplo[1]))
        aux = cuadruplo[2].split('#')
        if (aux[0] != '0' and aux[1] != '0'): #Si vienen 0
significa que es todo el arreglo
            de = int(aux[0][1:])
            a = int(aux[1][:])

```

```

        # Si es -1, significa que es 0
        if (aux[0] == -1):
            de = 0
        if (aux[1] == -1):
            a = 0

        if verificar(arreglo, de, a): # Verificar con cualquier
arreglo, ya que deben de ser del mismo tamaño
            for r in range(de,a+1):
subArreglo.append(arreglo[r]) #Crea arreglo a partir de los datos
sacados

            try:
                llenarValor(pilaCorriendo, cuadruplo[3], tipo,
statistics.variance(subArreglo))
            except:
                print("Error Maquina Virtual: Error al calcular
la varianza, del rango de {} a {}".format(de, a))
                sys.exit()
                return

        else:
            try:
                llenarValor(pilaCorriendo, cuadruplo[3], tipo,
statistics.variance(arreglo))
            except:
                print("Error Maquina Virtual: Error al calcular la
varianza, en todo el archivo")
                sys.exit()
                return

# correlation
elif cuadruplo[0] == 'correlacion':
    subArreglo1 = []
    subArreglo2 = []
    cuadAUX = cuadruplo[1].split('#')
    tipo1 = getTipo(cuadAUX[0][1:])
    tipo2 = getTipo(cuadAUX[1][:])
    arr1 = getValor(pilaCorriendo, cuadAUX[0][1:],tipo1)
    arr2 = getValor(pilaCorriendo, cuadAUX[1][:],tipo2)
    if (len(arr1) == len(arr2)):
        aux = cuadruplo[2].split("#")
        if (aux[0] != '0' and aux[1] != '0'): #Si vienen 0
significa que es todo el arreglo

```

```

        de = int(aux[0][1:])
        a = int(aux[1][:])
        # Si es -1, significa que es 0
        if (aux[0] == -1):
            de = 0
        if (aux[1] == -1):
            a = 0
        if verificar(arr1, de, a): # Verificar con
cualquier arreglo, ya que deben de ser del mismo tamaño
            for r in range(de,a+1): #Crea arreglos a partir
de los datos sacados
                subArreglo1.append(arr1[r])
                subArreglo2.append(arr2[r])
            try:
                i = np.corrcoef(subArreglo1,subArreglo2)
            except:
                print("Error Maquina Virtual: Error al
calcular la varianza, del rango de {} a {}".format(de, a))
                sys.exit()
                return
        else:
            try:
                i = np.corrcoef(arr1, arr2)
            except:
                print("Error Maquina Virtual: Error al calcular
la correlacion")
                sys.exit()
                return
        llenarValor(pilaCorriendo, cuadruplo[3],
getTipo(cuadruplo[3]), i[0][1])

    else:
        print("Error Maquina Virtual: Los dataframes para el
coeficiente de correlacion no son del mismo tamaño")
        # plothist
        elif cuadruplo[0] == 'histograma':
            subArreglo = []
            arr = getValor(pilaCorriendo, cuadruplo[1],
getTipo(cuadruplo[1]))

```

```

        if (cuadruplo[2] != '0' and cuadruplo[3] != '0'): #Si
vienen 0 significa que es todo el arreglo
            de = int(cuadruplo[2])
            a = int(cuadruplo[3])
            # Si es -1, significa que es 0
            if (cuadruplo[2] == -1):
                de = 0
            if (cuadruplo[3] == -1):
                a = 0
            if verificar(arr, de, a): # Verificar con cualquier
arreglo, ya que deben de ser del mismo tamaño
                for r in range(de,a+1):
                    subArreglo.append(arr[r]) #Crea arreglo a partir
de los datos sacados
                try:
                    a = np.histogram(subArreglo)
                    plt.hist(a)
                    plt.savefig("histograma.png")
                    plt.show()
                    print("WARNING Maquina Virtual: En caso de que
no se vea la imagen, se guardo como histograma.png")
                    plt.close()
                except:
                    print("Error no se logro graficar el
histograma")
                    sys.exit()
            else:
                try:
                    a = np.histogram(arr)
                    plt.hist(a)
                    plt.savefig("histograma.png")
                    plt.show()
                    print("WARNING Maquina Virtual: En caso de que no
se vea la imagen, se guardo como histograma.png")
                    plt.close()
                except:
                    print("Error no se logro graficar el histograma")
                    sys.exit()

# plotline
elif cuadruplo[0] == 'plotline':

```



```

        subArreglo1 = []
        subArreglo2 = []
        arr1 = getValor(pilaCorriendo, cuadruplo[1],
getTipo(cuadruplo[1]))
        arr2 = getValor(pilaCorriendo, cuadruplo[2],
getTipo(cuadruplo[2]))
        axisX = [] # Para tener donde comporar los arreglos
        if (len(arr1) == len(arr2)): # Dataframes deben ser del
mismo tamaño
            if (cuadruplo[2] != '0' and cuadruplo[3] != '0'): #Si
vienen 0 significa que es todo el arreglo
                de = 0 # Siempre sera de 0 en adelante
                a = int(getValor(pilaCorriendo, cuadruplo[3],
getTipo(cuadruplo[3])))
                if (a == -1):
                    a = 0
                if verificar(arr1, de, a): # Verificar con
cualquier arreglo, ya que deben de ser del mismo tamaño
                    for r in range(de,a+1): #Crea arreglos a partir
de los datos sacados
                        subArreglo1.append(arr1[r])
                        subArreglo2.append(arr2[r])
                        axisX.append(r)
                    try:
                        plt.plot(axisX,subArreglo1,
label='Dataframe 1')
                        plt.plot(axisX,subArreglo2,
label='Dataframe 2')
                        plt.legend()
                        plt.savefig("plotline.png")
                        plt.show()
                        print("WARNING Maquina Virtual: En caso de
que no se vea la imagen, se guardo como plotline.png")
                        plt.close()
                    except:
                        print("Error no se logro graficar")
                        sys.exit()
            else:
                #Create x Axis
                for r in range(0, len(arr1) - 1):axisX.append(r)

```

```

        try:
            plt.plot(axisX, arr1, label='Dataframe 1')
            plt.plot(axisX, arr2, label='Dataframe 2')
            plt.legend()
            plt.savefig("plotline.png")
            plt.show()
            print("WARNING Maquina Virtual: En caso de que
no se vea la imagen, se guardo como plotline.png")
            plt.close()
        except:
            print("Error no se logro graficar")
            sys.exit()

    else:
        print("Error Maquina Virtual: Los dataframes en
plotline no son del mismo tamaño")

#Carga de Archivos
elif cuadruplo[0] == 'carga':
    nombreArchivo = cuadruplo[2]
    nombreArchivo = nombreArchivo.replace('"', '')
    nombreArchivo = nombreArchivo.replace("'", '')
    archivo = getArchivo(nombreArchivo)
    arreglo = []
    for val in archivo:
        try:
            int(val)
            val = int(val)
        except:
            try:
                float(val)
                val = float(val)
            except:
                pass
        arreglo.append(val)
    llenarValor(mem_GLOBAL, cuadruplo[1],
getTipo(cuadruplo[1]), arreglo)

#FINPROGRAMA
elif cuadruplo[0] == 'FINPROGRAMA':
    terminado = True

# OPERADORES
elif cuadruplo[0] != 'CONS':

```

```
operadores(cuadroplo[0])

# Controla el indice para saber que cuadroplo ejecutar
if sigCuaIndice != -1:
    cuaIndice = sigCuaIndice
else:
    cuaIndice = cuaIndice + 1
```

7. Manual de Usuario

El manual de usuario se encuentra en el repositorio de Github como el archivo de README.md:

<https://github.com/rodvama/compilers-final>

Video demo:

<https://youtu.be/kZ1QVeUIW5o>