

**Diseño de Compiladores**

**Grupo 1**

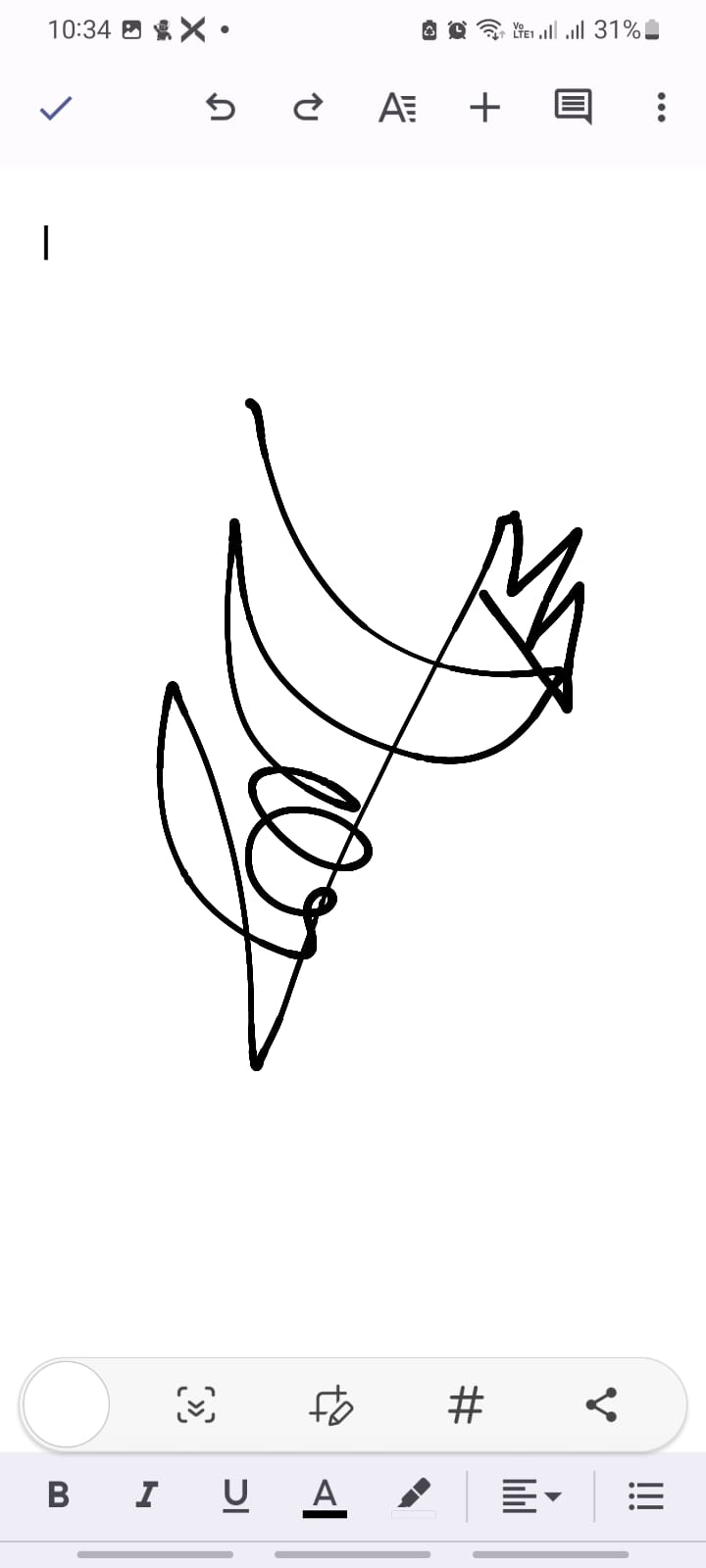
**Profesores**

Elda Guadalupe Quiroga Gonzales

Hector Gibran Ceballos Cancino

**Gallagher Compillator**

A01231795 Diego Garza Macias



5 Junio 2023

**INDEX**

[**1 Acerca del Proyecto 1**](#_25dfbnt9rpmm)

[1.1 Propósito Y Alcance del Proyecto 1](#_r3sdbqu3oiuj)

[1.2 Análisis de Requerimientos 1](#_viizz9os6uxt)

[1.2.1 Principales Casos de Prueba 2](#_3e92xxmfr4wy)

[**1.3 Proceso de Desarrollo 3**](#_djr7s5yl381m)

[**1.3.1 Commits Repositorio GitHub 4**](#_8ne1n4k9r81c)

[**1.3.2 Reflexión 14**](#_ep51mji2lv0r)

[**2 El Lenguaje 15**](#_nlfilkp9uv4f)

[**2.1 Nombre y origen 15**](#_q3uarpl8yimo)

[**2.2 Descripción del Lenguaje 15**](#_jlgvwoud5vwl)

[**2.3 Lista de Errores Posibles 15**](#_turlkhpedvrn)

[**3 Descripción del Compilador 16**](#_1tz19rxol8jb)

[**3.1 Equipo de Computo, lenguaje y librerías utilizadas 16**](#_agn6tdl0n5s3)

[**3.2 Descripción Análisis Léxico 16**](#_jj92qi6xonug)

[**3.3 Sintaxis gramática formal 18**](#_oek4iro3l260)

[3.4 Generación de Código Intermedio y Análisis Semántico 22](#_111hosyz4u1r)

[**3.5 Memoria De Compilación 23**](#_s3ajaekd751u)

[**3.6 Tipos De Operadores 24**](#_wxcjc461r7tt)

[**3.7 Diagramas de flujo 25**](#_7yig3mqnudzr)

[**3.8 Cubo Semántico 29**](#_du8wo1qeei6s)

[**3.8 Memoria Virtual 36**](#_sr8gjsebpfia)

# 1 Acerca del Proyecto

## 1.1 Propósito Y Alcance del Proyecto

El propósito de este proyecto fue el de crear un compilador propio del autor. Para esto fue necesario que este utilizara todos los conocimientos recaudados no solo durante la clase actual si no de toda la carrera cursada por este mismo. Para lograr utilizar el compilador fue necesario que este haya sido implementado en una máquina virtual la cual permitirá hacer función de este compilador, el cual lleva por nombre Gallagher nombre por el cual será mencionado a partir de este punto .

## 1.2 Análisis de Requerimientos

A continuación se presentan los requerimientos que deberán ser cumplidos por Gallagher y su máquina virtual para que estos sean considerados un éxito.

1. Gallagher debe ser capaz de leer un archivo proporcionado por el usuario con el código escrito por el usuario.
2. Una vez leído el archivo Gallagher debe de ser capaz de reproducir este código en forma de cuádruplos.
3. Los cuádruplos deben de ser almacenados para después alimentar a la máquina virtual para que esta pueda interpretarlos para el usuario.
4. En caso de que Gallagher se encuentre con algún error de escritura sobre el código durante la lectura de este, Gallagher debe proporcionar un mensaje el cual explique al usuario porque éste tuvo un error.
5. Gallagher debe de ser capaz de interpretar operaciones aritméticas (exponente, multiplicación, división, suma, resta) además de respetar la jerarquía de estas (PEMDAS)
6. Gallagher deber ser capaz de asignar estos resultados a variables las cuales serán inicializadas con el operador *“let”*
7. Gallagher debe de tener la capacidad de comparar dos valores con diferentes operadores para después tomar decisiones en base a estos (If, else)
8. Gallagher debe ser capaz de entrar a un estado cíclico hasta que cierto valor cumpla con un requerimiento específico.
9. Gallagher debe de ser capaz de agrupar estas funcionalidades en forma de módulos los cuales deberán de ser declarados con la palabra func
10. Además de guardar la información, Gallagher debe de ser capaz de guardar está en un formato de un vector de N dimensiones.
11. Gallagher debe de ser capaz de acceder a cualquier parte de este Vector de N dimensiones.
12. La máquina virtual debe de ser capaz de leer las instrucciones generadas por Gallagher de una forma eficiente.
13. La máquina virtual deberá de mantener su memoria asignada dependiendo de las direcciones creadas por Gallagher.
14. La máquina virtual y Gallagher deben de complementarse para que estos puedan lograr saltos los cuales puedan llevar a la recursión.
15. Al Gallagher estar enfocado en un público con interés en estadística este debe de ser capaz de diferentes funcionalidades como lo son
    1. Media
    2. Mediana
    3. Moda
    4. Graficar

## 1.2.1 Principales Casos de Prueba

A continuacion se dara un listado de los principales casos de prueba para que Gallagher sea considerado un éxito:

1. Secuencia Fibonacci por recursión.
2. Encontrar el índice de un valor en un vector.
3. Gallagher debe de ser capaz de sacar promedio de un vector.
4. Gallagher debe de ser capaz de realizar cualquier operación matemática que ocupe que utilice los símbolos aceptados por este.

# 1.3 Proceso de Desarrollo

Para la realización de este proyecto fue necesario que fuera dividido por varias entregas hasta alcanzar el punto actual. Las entregas fueron establecidas y divididas por los educadores de la siguiente manera.

1. Avance 0 Tokens Diagramas de Sintaxis y Gramática Entrega ( 23 de Abri 2023 )

Para esta entrega el autor decidió el enfoque se le daría al proyecto, además de tomar la decisión de crear este mismo de manera individual. Algunos ejemplos de funcionalidades que se deseo incluir en Gallagher fueron investigadas en esta fase.

1. Avance 1 Lexico Sintaxis ( Abril 16 2023)

Para esta entrega se comenzó a decidir qué palabras serían reservadas para el desarrollo de Gallagher. Además se decidió el motor léxico utilizado para este proyecto.

1. Avance 2 Semántica Básica de Variables y Cubo Semántico (Abril 23 2023)

Para esta entrega el autor comenzó a investigar la semántica que llevaría Gallagher para su desarrollo.

1. Avance 3 Semántica Generacion de Codigo de Expresiones y estatutos lineales (Abril 30 2023)

Para esta entrega se buscó aprender sobre el generador léxico y el parser que fue utilizado para la creación del compilador.

1. Avance 4 Generación de Código y Estatuos Condicionales y Cíclicos (Mayo 7 2023)

Para esta entrega se estableció el formato de archivos para el compilador.

1. Avance 5 Generacion de Codigos de Funciones (Mayo 14 2023)

Para esta entrega se comenzó a trabajar en los primeros formatos de cubo semántico.

1. Avance 6 Mapa de Memoria de ejecución para la Maquina Virtual y Ejecución de expresiones (Mayo 22 2023)

Se creó el cubo semántico, se crearon los regex y diccionario de palabras necesarios para el compilador. Se comenzó a trabajar en la gramática.

1. Avance 7 Codigo de arreglos y ejecución (Mayo 29 2023)

Avance mayor mente en la creación de la gramática para el compilador, creación de diferentes útiles para el uso del programa. Cuadrantes de operaciones matemáticas y condicionales listos.

1. Avance 8 Código y Ejecución Aplicación particular (Junio 3 2023)

Creación de máquina virtual para el compilador Gallagher, terminación de cuadros para ciclo while, creación de vector de N dimensiones, creación de diferentes estatus para la compilación.

Por diferentes cuestiones personales el autor del compilador realizó varias fases de la creación del compilador en desfase. Siempre buscando crear el mejor compilador posible.

# 1.3.1 Commits Repositorio GitHub



commit ff97c484fcc9b19783680ac8ba8e2f082583ad57 (HEAD -> gramatica, origin/gramatica)

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 09:18:39 2023 -0600

Fix while

commit 46ef73a278020be38c275f6da8aa7741849d0824

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 02:17:23 2023 -0600

:...skipping...

commit ff97c484fcc9b19783680ac8ba8e2f082583ad57 (HEAD -> gramatica, origin/gramatica)

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 09:18:39 2023 -0600

Fix while

commit 46ef73a278020be38c275f6da8aa7741849d0824

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 02:17:23 2023 -0600

One array At the time,

lets get more than one

commit 85b5b12fa3b6d168d74ff511bc39110c1cc8719c

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 01:29:13 2023 -0600

Safe point N array getter

commit 544fd739508d5a876e9cbac88f606fd4946aa23d

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sun Jun 4 12:39:50 2023 -0600

N Array creation

commit 91bf4cff32b78c47b8d47f951ed3921ba518be51

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sun Jun 4 06:50:32 2023 -0600

RECURSIVITY WORKS LGGGGGGGGGGGGGGGGGGGGG

commit a6de603547ccd31611b955aabdaad2ad0bb3c7d1

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 23:48:17 2023 -0600

Debuggin memories

commit b6afdbc0cc04201fa65fa96715f8a1568f64c6bd

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 22:16:23 2023 -0600

delete wrong import

commit 44d5ac9cf786aefcb478b7e613417109f34ec039

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 22:15:31 2023 -0600

Execution memory almost working

i need to fix jumps

i was hoping this moment didnt arrived

commit cf039531b292465bbbd5492dfdff234a06fb6392

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 15:45:47 2023 -0600

Variable function assign work

commit 3121c93dbf8b4c3b11975f369327934cf123f968

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Fri Jun 2 20:41:33 2023 -0600

Virtual memory semiworking

commit 9419dbd2dd4186724536aa10b59ed360d0a93927

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Fri Jun 2 14:07:05 2023 -0600

QUAD CORRECTION

NEED TO FIX JUMP VALUES

START VIRTUAL MEMORY

commit 3583cfa5fbef73186687a8eefdad6bf0c9c6bf12

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 15:18:13 2023 -0600

Start func calling

commit 171b67d83a48653b04c34156f22b89938cbc03fc

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 12:10:05 2023 -0600

While working

still need to check jumps lol

commit 7a25ea12bd9e61e8892232cc2ff2abbc138aa5fa

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 10:14:42 2023 -0600

Change Inital Quad

commit e02dd5f56c724fd46d3a9a556a74017eeb726829

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 08:56:37 2023 -0600

If else working

gotta fix some jumps

else if dont know

commit b5280085554a5cfff2504389bda888c196e88a73

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Tue May 30 03:20:14 2023 -0600

ASIGNACION WORKING

commit a7c62e0faa03b11a7126275b503a519dd9287017

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Tue May 30 01:39:37 2023 -0600

PEMDAS DONE

commit 1bb4dfcde72dd47506378e55faffca3a13ca0452 (main)

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 23:28:18 2023 -0600

EMDAS ready

come at me P

commit 2d87170c07a7ea164840951bdfd73fadada28609

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 22:52:35 2023 -0600

MDAS Works

start to work on PE

commit 43fbb41cce4edc3e465ecc773f39235dbf4c74dc

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 21:31:50 2023 -0600

+ and - stacks working wooo

commit a3d0d11e7906b7808421ce6e1ac6eb30f7d5e215

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 11:56:17 2023 -0600

SET quad generator Working

commit cbd88e8e3a586d9aaa0120c072b73939bf3e2ed4

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 03:35:43 2023 -0600

Assign works?? Maybe I dont know cant you repeat the question

commit d18de5cfb096b700e82f65dc2d4050aa70d67aea

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sun May 28 20:44:53 2023 -0600

Global variables working

commit ff97c484fcc9b19783680ac8ba8e2f082583ad57 (HEAD -> gramatica, origin/gramatica)

commit ff97c484fcc9b19783680ac8ba8e2f082583ad57 (HEAD -> gramatica, origin/gramatica)

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 09:18:39 2023 -0600

Fix while

commit 46ef73a278020be38c275f6da8aa7741849d0824

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 02:17:23 2023 -0600

One array At the time,

lets get more than one

commit 85b5b12fa3b6d168d74ff511bc39110c1cc8719c

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 01:29:13 2023 -0600

Safe point N array getter

commit 544fd739508d5a876e9cbac88f606fd4946aa23d

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sun Jun 4 12:39:50 2023 -0600

N Array creation

commit 91bf4cff32b78c47b8d47f951ed3921ba518be51

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sun Jun 4 06:50:32 2023 -0600

RECURSIVITY WORKS LGGGGGGGGGGGGGGGGGGGGG

commit a6de603547ccd31611b955aabdaad2ad0bb3c7d1

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 23:48:17 2023 -0600

Debuggin memories

commit b6afdbc0cc04201fa65fa96715f8a1568f64c6bd

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 22:16:23 2023 -0600

delete wrong import

commit 44d5ac9cf786aefcb478b7e613417109f34ec039

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 22:15:31 2023 -0600

Execution memory almost working

i need to fix jumps

i was hoping this moment didnt arrived

commit cf039531b292465bbbd5492dfdff234a06fb6392

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 15:45:47 2023 -0600

Variable function assign work

commit 3121c93dbf8b4c3b11975f369327934cf123f968

Author: DiegoGarza <diego@pidedirecto.mx>

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Fri Jun 2 20:41:33 2023 -0600

Virtual memory semiworking

commit 9419dbd2dd4186724536aa10b59ed360d0a93927

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Fri Jun 2 14:07:05 2023 -0600

QUAD CORRECTION

NEED TO FIX JUMP VALUES

START VIRTUAL MEMORY

commit 3583cfa5fbef73186687a8eefdad6bf0c9c6bf12

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 15:18:13 2023 -0600

Start func calling

commit 171b67d83a48653b04c34156f22b89938cbc03fc

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 12:10:05 2023 -0600

While working

still need to check jumps lol

commit 7a25ea12bd9e61e8892232cc2ff2abbc138aa5fa

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 10:14:42 2023 -0600

Change Inital Quad

commit e02dd5f56c724fd46d3a9a556a74017eeb726829

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 08:56:37 2023 -0600

If else working

gotta fix some jumps

else if dont know

commit b5280085554a5cfff2504389bda888c196e88a73

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Tue May 30 03:20:14 2023 -0600

ASIGNACION WORKING

commit a7c62e0faa03b11a7126275b503a519dd9287017

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Tue May 30 01:39:37 2023 -0600

PEMDAS DONE

commit 1bb4dfcde72dd47506378e55faffca3a13ca0452 (main)

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 23:28:18 2023 -0600

EMDAS ready

come at me P

commit 2d87170c07a7ea164840951bdfd73fadada28609

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 22:52:35 2023 -0600

MDAS Works

start to work on PE

commit 43fbb41cce4edc3e465ecc773f39235dbf4c74dc

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 21:31:50 2023 -0600

+ and - stacks working wooo

commit a3d0d11e7906b7808421ce6e1ac6eb30f7d5e215

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 11:56:17 2023 -0600

SET quad generator Working

commit cbd88e8e3a586d9aaa0120c072b73939bf3e2ed4

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 03:35:43 2023 -0600

Assign works?? Maybe I dont know cant you repeat the question

commit d18de5cfb096b700e82f65dc2d4050aa70d67aea

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sun May 28 20:44:53 2023 -0600

Global variables working

commit bd3e5d64d15c582d59ee2d0f5efa25b8faf037e0

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sun May 28 20:21:10 2023 -0600

Global variables working

commit 43fbb41cce4edc3e465ecc773f39235dbf4c74dc

commit ff97c484fcc9b19783680ac8ba8e2f082583ad57 (HEAD -> gramatica, origin/gramatica)

commit ff97c484fcc9b19783680ac8ba8e2f082583ad57 (HEAD -> gramatica, origin/gramatica)

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 09:18:39 2023 -0600

Fix while

commit 46ef73a278020be38c275f6da8aa7741849d0824

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 02:17:23 2023 -0600

One array At the time,

One array At the time,

lets get more than one

commit 85b5b12fa3b6d168d74ff511bc39110c1cc8719c

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 01:29:13 2023 -0600

Safe point N array getter

commit 544fd739508d5a876e9cbac88f606fd4946aa23d

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sun Jun 4 12:39:50 2023 -0600

N Array creation

commit 91bf4cff32b78c47b8d47f951ed3921ba518be51

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sun Jun 4 06:50:32 2023 -0600

RECURSIVITY WORKS LGGGGGGGGGGGGGGGGGGGGG

commit a6de603547ccd31611b955aabdaad2ad0bb3c7d1

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 23:48:17 2023 -0600

Debuggin memories

commit b6afdbc0cc04201fa65fa96715f8a1568f64c6bd

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 22:16:23 2023 -0600

delete wrong import

commit 44d5ac9cf786aefcb478b7e613417109f34ec039

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 22:15:31 2023 -0600

Execution memory almost working

i need to fix jumps

i was hoping this moment didnt arrived

commit cf039531b292465bbbd5492dfdff234a06fb6392

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sat Jun 3 15:45:47 2023 -0600

Variable function assign work

commit 3121c93dbf8b4c3b11975f369327934cf123f968

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Fri Jun 2 20:41:33 2023 -0600

Virtual memory semiworking

commit 9419dbd2dd4186724536aa10b59ed360d0a93927

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Fri Jun 2 14:07:05 2023 -0600

QUAD CORRECTION

NEED TO FIX JUMP VALUES

START VIRTUAL MEMORY

commit 3583cfa5fbef73186687a8eefdad6bf0c9c6bf12

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 15:18:13 2023 -0600

Start func calling

commit 171b67d83a48653b04c34156f22b89938cbc03fc

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 12:10:05 2023 -0600

While working

still need to check jumps lol

commit 7a25ea12bd9e61e8892232cc2ff2abbc138aa5fa

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 10:14:42 2023 -0600

Change Inital Quad

commit e02dd5f56c724fd46d3a9a556a74017eeb726829

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Wed May 31 08:56:37 2023 -0600

If else working

gotta fix some jumps

else if dont know

commit b5280085554a5cfff2504389bda888c196e88a73

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Tue May 30 03:20:14 2023 -0600

ASIGNACION WORKING

commit a7c62e0faa03b11a7126275b503a519dd9287017

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Tue May 30 01:39:37 2023 -0600

PEMDAS DONE

commit 1bb4dfcde72dd47506378e55faffca3a13ca0452 (main)

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 23:28:18 2023 -0600

EMDAS ready

come at me P

commit 2d87170c07a7ea164840951bdfd73fadada28609

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 22:52:35 2023 -0600

MDAS Works

start to work on PE

commit 43fbb41cce4edc3e465ecc773f39235dbf4c74dc

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 21:31:50 2023 -0600

+ and - stacks working wooo

commit a3d0d11e7906b7808421ce6e1ac6eb30f7d5e215

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 11:56:17 2023 -0600

SET quad generator Working

commit cbd88e8e3a586d9aaa0120c072b73939bf3e2ed4

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon May 29 03:35:43 2023 -0600

Assign works?? Maybe I dont know cant you repeat the question

commit d18de5cfb096b700e82f65dc2d4050aa70d67aea

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sun May 28 20:44:53 2023 -0600

Global variables working

commit bd3e5d64d15c582d59ee2d0f5efa25b8faf037e0

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Sun May 28 20:21:10 2023 -0600

commit ff97c484fcc9b19783680ac8ba8e2f082583ad57 (HEAD -> gramatica, origin/gramatica)

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 09:18:39 2023 -0600

Fix while

commit 46ef73a278020be38c275f6da8aa7741849d0824

Author: DiegoGarza <diego@pidedirecto.mx>

Date: Mon Jun 5 02:17:23 2023 -0600

One array At the time,

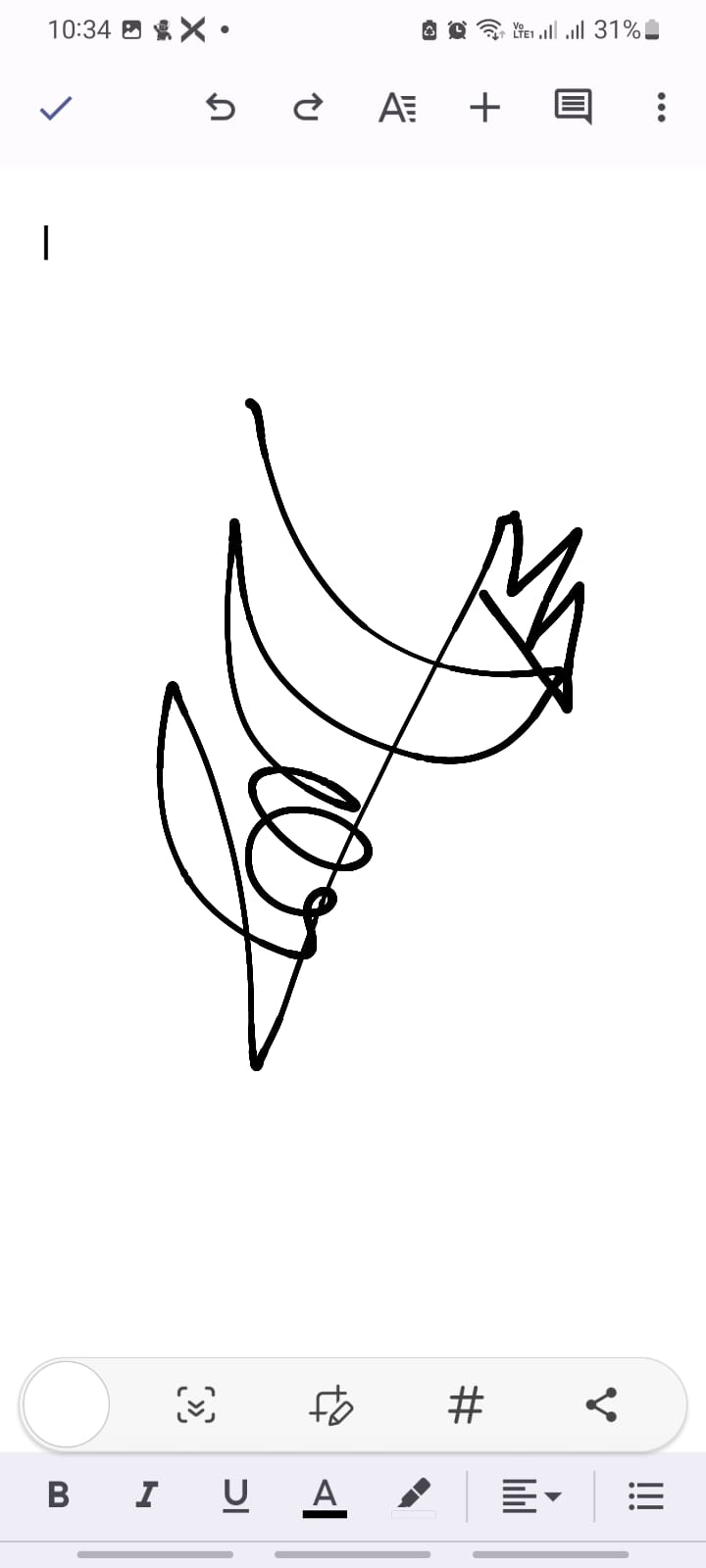
lets get more than one

:



# 1.3.2 Reflexión

Durante la realización de este proyecto el autor aprendió diferentes lecciones tanto de la materia como de carácter personal. El ver atrás diferentes conocimientos de la carrera e implementarlos durante el desarrollo de este compilador le trajo al autor diferentes sentimientos. Además de implementar conocimientos adquiridos fuera del aula de clase en diferentes pasantías y trabajos de tiempo completo la organización de código de cierta forma, el uso de github y diferentes técnicas fueron algo que el autor disfruto implementar. Diferentes complicaciones de carácter personal afectaron la distribución de trabajo, pero el autor siempre busco lograr hacer el mejor compilador posible.



# 2 El Lenguaje

## 2.1 Nombre y origen

El lenguaje fue nombrado Gallagher, en honor a la serie que acompañó al autor durante el desarrollo de este proyecto. Además de la enseñanza de que por más mal que se ve algo se puede salir de la situación.

## 2.2 Descripción del Lenguaje

Gallagher es un lenguaje que se lee de forma vertical. Este posee diferentes características y funcionalidades como lo son:

* Creación de variables de diferentes tipos int, float, string empezando con la palabra reservada let
* Salida de información a través del comando printG
* Operación aritméticas respetando la jerarquía de operaciones.
* Creación de vector de N dimensiones
* Condicionales con estatutos IF y Else
* Ciclos con estatuto while
* Creación de funciones para reutilización de código
* Capacidad de utilizar funciones de manera recursiva.

## 2.3 Lista de Errores Posibles

* Uso de caracteres no reconocidos.
* Uso indebido de variables.
* Cantidad errónea de parámetros en funciones.
* Uso erróneo de paréntesis.
* Uso indebido de palabras reservadas.

# 3 Descripción del Compilador

## 3.1 Equipo de Computo, lenguaje y librerías utilizadas

Gallagher fue desarrollado en el sistema operativo macOs Ventura, en el IDLE PyCharm.

El lenguaje de programación utilizado

Se utilizó la librería PLY para el parser y lexer el cual fue instalado con PIP.

Para el sistema de versiones se utilizó Git y plataforma en linea Github.

## 3.2 Descripción Análisis Léxico

Tokens Válidos en el compilador Gallagher

| Token | Regex |
| --- | --- |
| INI\_INT | r’\d+’ |
| INI\_FLOAT | r’\d+\.\d\_+’ |
| INI\_STRING | r’”([^”\n]|(\\”))\*”$’ |
| ID | r’[a-zA-Z\_][a-zA-Z\_0-9]\*’ |
| IGNORE | r’\s’ |
| ADD | r’\+’ |
| LESS | r’\-’ |
| SPLIT\_BY | r’\/’ |
| MULT\_BY | r’\\*’ |
| TIMES\_BY\_SAME | r’\^’ |
| SAME | r’\==’ |
| SET | r’\=’ |
| COMMA | r’\,’ |
| DOTCOMMA | r’\;’ |
| TWOPOINTS | r’\:’ |
| LEFTPARENT | r’\(’ |
| RIGHTPARENT | r’)’ |
| LEFTBRACK | r’\{’ |
| RIGHTBRACK | r’\}’ |

| Token | Regex |
| --- | --- |
| LEFTKEY | r’\{’ |
| RIGHTKEY | r’\}’ |
| LESSTN | r’\<’ |
| GREATERTN | r’\>’ |
| NOTSAME | r’\!=’ |
| IGNORE | r’[a-zA-Z\_][a-zA-Z\_0-9]\*’ |
| GOTOFUNC | r’[a-zA-Z\_][a-zA-Z\_0-9]\*’ |
| GOTO | r’[a-zA-Z\_][a-zA-Z\_0-9]\*’ |
| GOTOT | r’[a-zA-Z\_][a-zA-Z\_0-9]\*’ |
| GOTOF | r’[a-zA-Z\_][a-zA-Z\_0-9]\*’ |
| ENDFUNC | r’[a-zA-Z\_][a-zA-Z\_0-9]\*’ |
| ENDALL | r’[a-zA-Z\_][a-zA-Z\_0-9]\*’ |
| AT | r’\@’ |
| LESSSAME | r’\<=’ |
| MORESAME | r’\>=’ |

Palabras Reservadas

| Palabra | Token |
| --- | --- |
| Programa | PROG |
| Main | MAIN |
| func | FUNC |
| let | LET |
| if | IF |
| else | ELSE |
| int | INT |
| float | FLOAT |
| bool | BOOL |

| void | VOID |
| --- | --- |
| printG | PRINTG |
| do | DO |
| while | WHILE |
| return | RETURN |

## 3.3 Sintaxis gramática formal

Sintaxis definida para encontrar tokens y palabras reservadas

"""

compile : PROG ID seen\_program DOTCOMMA lets modules

|

"""

""" lets : LET seen\_lets type ID seen\_ID\_let aux\_let DOTCOMMA lets

| LET seen\_lets type ID seen\_ID\_let LEFTBRACK left\_bracket\_array add\_dimension RIGHTBRACK end\_array\_init DOTCOMMA lets

| empty

"""

"""

type : INT seen\_type

| FLOAT seen\_type

| BOOL seen\_type

"""

"""

access\_array : ID LEFTBRACK access\_array\_found array\_expresion RIGHTBRACK end\_array\_access

"""

"""

aux\_let : COMMA ID seen\_ID\_let aux\_let

|

"""

"""

func : FUNC ID seen\_func\_name params TWOPOINTS return\_func\_type TWOPOINTS func\_code

"""

'''

params : LEFTPARENT param\_table\_init param\_declare RIGHTPARENT

'''

'''

return\_func\_type : type

| VOID void\_detect

'''

"""

func\_code : LEFTKEY func\_code\_start lets func\_code\_aux RIGHTKEY end\_func

"""

"""

action : assign

| expresion\_line

| condition

| while

| func\_call DOTCOMMA

| write

| return

"""

"""

return : RETURN expresion\_line return\_aux DOTCOMMA

|

"""

"""

write : PRINTG LEFTPARENT write\_found RIGHTPARENT DOTCOMMA

"""

"""

write\_found\_aux : expresion\_line print\_exp

| INI\_STRING string\_appear print\_exp

"""

"""

func\_call : AT ID func\_call\_ID LEFTPARENT func\_calls\_params end\_func\_call\_params RIGHTPARENT end\_func\_call

|

"""

"""

while : WHILE while\_appear LEFTPARENT condition\_expresion RIGHTPARENT right\_parent\_condition condition\_code while\_end

"""

"""

condition : IF LEFTPARENT found\_init\_parent condition\_expresion right\_parent\_condition RIGHTPARENT condition\_code condition\_end\_check

"""

"""

condition\_end\_check : ELSE else\_appear condition\_code end\_condition

| end\_condition

"""

"""

condition\_code : LEFTKEY lets func\_code\_aux RIGHTKEY

"""

"""

condition\_signs : GREATERTN add\_operator

| LESSTN add\_operator

| SAME add\_operator

| NOTSAME add\_operator

| LESSSAME add\_operator

| MORESAME add\_operator

"""

"""

aux\_expresion : ADD add\_operator expresion\_line

| LESS add\_operator expresion\_line

"""

"""

aux\_term : SPLIT\_BY add\_operator term

| MULT\_BY add\_operator term

"""

"""

parent\_aux : LEFTPARENT found\_init\_parent expresion\_line RIGHTPARENT found\_end\_parent

"""

"""

aux\_expo : TIMES\_BY\_SAME add\_operator fact

"""

"""

add\_dimension : INI\_INT array\_start INI\_INT array\_end

| INI\_INT array\_start INI\_INT array\_end COMMA extra\_dimension\_found add\_dimension

"""

"""

call\_let : ID check\_let\_exists

"""

"""

call\_lets : INI\_INT check\_global\_const\_exists

| INI\_FLOAT check\_global\_const\_exists

"""

"""

set\_value : INI\_INT aux\_int\_check append\_operand DOTCOMMA

| INI\_FLOAT aux\_float\_check append\_operand DOTCOMMA

"""



## 3.4 Generación de Código Intermedio y Análisis Semántico

El compilador Gallagher crea un código intermedio de carácter de cuádruplos. Para lograr esto cada operación necesaria es representada por un array de 4 posiciones donde la primera representa la acción ( salto, operación, comparación, etc). Además de que existe un contador en memoria para saber cuándo y a donde es necesario saltar.

Estos cuadruplos empiezan con un [] para emparejar el contador con la asignación de cuadrupos.

Aqui un ejemplo de codigo intermedio con vectores:

[[],

['GOTO', '', '', 2],

['=', 6001, '', 2001],

['VERIFY', 6002, 6003, 6004],

['\*', 6002, 6005, 4001],

['VERIFY', 6006, 6003, 6006],

['\*', 6006, 6003, 4002],

['VERIFY', 6006, 6003, 6004],

['+', 6006, 4002, 4003],

['+', 4003, 6007, 4004],

['+', 4004, 6008, 4005],

['+', 6003, 2001, 4006],

['=', 4006, '', '\*4005'],

['VERIFY', 6003, 6003, 6004],

['\*', 6003, 6005, 4007],

['VERIFY', 6006, 6003, 6006],

['\*', 6006, 6003, 4008],

['VERIFY', 6004, 6003, 6004],

['+', 6004, 4008, 4009],

['+', 4009, 6007, 4010],

['+', 4010, 6008, 4011],

['=', 6002, '', '\*4011'],

['VERIFY', 6002, 6003, 6004],

['\*', 6002, 6005, 4012],

['VERIFY', 6006, 6003, 6006],

['\*', 6006, 6003, 4013],

['VERIFY', 6006, 6003, 6004],

['+', 6006, 4013, 4014],

['+', 4014, 6007, 4015],

['+', 4015, 6008, 4016],

['PRINTG', '', '', '\*4016'],

['VERIFY', 6003, 6003, 6004],

['\*', 6003, 6005, 4017],

['VERIFY', 6006, 6003, 6006],

['\*', 6006, 6003, 4018],

['VERIFY', 6004, 6003, 6004],

['+', 6004, 4018, 4019],

['+', 4019, 6007, 4020],

['+', 4020, 6008, 4021],

['PRINTG', '', '', '\*4021'],

['ENDALL', '', '', '']]



## 3.5 Memoria De Compilación

Es necesario que la memoria de compilación este dividida en diferentes cuadrantes, esto para su facil acceso y validación de datos ya sea por tipo o por valor en el scope (‘loca’, ‘temporal’, ‘global’)

memorySizes = {

'GLOBAL\_LET\_START': 0,

'GLOBAL\_LET\_INT\_START': 0,

'GLOBAL\_LET\_INT\_END': 500,

'GLOBAL\_LET\_FLOAT\_START': 501,

'GLOBAL\_LET\_FLOAT\_END': 1000,

'GLOBAL\_LET\_BOOL\_START': 1001,

'GLOBAL\_LET\_BOOL\_END': 1500,

'GLOBAL\_LET\_STRING\_START': 1501,

'GLOBAL\_LET\_STRING\_END': 2000,

'LOCAL\_LET\_START': 2001,

'LOCAL\_LET\_INT\_START': 2001,

'LOCAL\_LET\_INT\_END': 2500,

'LOCAL\_LET\_FLOAT\_START': 2501,

'LOCAL\_LET\_FLOAT\_END': 3000,

'LOCAL\_LET\_BOOL\_START': 3001,

'LOCAL\_LET\_BOOL\_END': 3500,

'LOCAL\_LET\_STRING\_START': 3501,

'LOCAL\_LET\_STRING\_END': 4000,

'TEMPORAL\_LET\_START': 4001,

'TEMPORAL\_LET\_INT\_START': 4001,

'TEMPORAL\_LET\_INT\_END': 4500,

'TEMPORAL\_LET\_FLOAT\_START': 4501,

'TEMPORAL\_LET\_FLOAT\_END': 5000,

'TEMPORAL\_LET\_BOOL\_START': 5001,

'TEMPORAL\_LET\_BOOL\_END': 5500,

'TEMPORAL\_LET\_STRING\_START': 5501,

'TEMPORAL\_LET\_STRING\_END': 6000,

'CONSTANT\_START': 6001,

'CONSTANT\_END': 8000

}



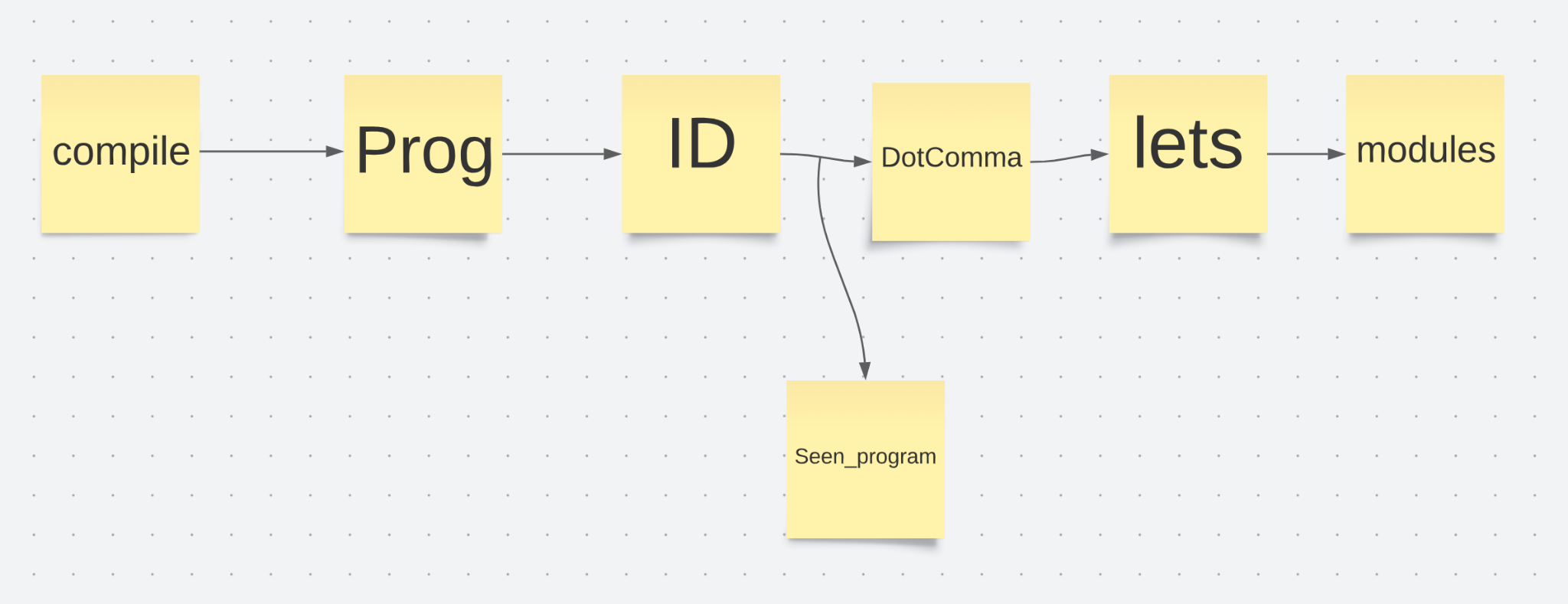
## 3.6 Tipos De Operadores

| Operador | Descripción |
| --- | --- |
| GOTO , ‘’, ’’, ’(to)‘ | La maquina al encontrar este tipo de operador realizará un salto automáticamente sin ningún requerimiento |
| * , ‘ ‘, ‘ ‘, ‘ ‘ (- ,^, \*, / ), = | Aritmeticos, nos sirven para realizar operaciones. |
| == , != , < , > , <= , >, | Booleano nos sirve para crear una comparación entre dos variables. |
| GOTOF , let ,’ ‘ , (to) | Salto condicional, la máquina cambia de cuadrante si el valor con posición 0 cumple el requerimiento |
| PrintG, ‘’ , ‘’, let | Imprime la variable o el valor en el último espacio |
| ENDFUNC, ‘’ ,’’ ,’’ | Nos sirve para saber si una función termina, y así desechar las variables temporales |
| GOSUB, funcName, ‘’, (to) | Nos sirve para saber que se esta llamando una función y que tenemos que hacer el salto al cuádruplo que indica |
| ERA, ‘’, ‘’, funcName | Cuadrante utilizado para liberar el espacio necesario en la función si esta es llamada |
| PARAM, let, ‘’ , paramCounter | Cuadrante utilizado para agregar un parametro a una función llamada |
| ENDPROG | nos sirve para saber que se acabo el progama |
| VERIFY | Nos sirve para saber si un vector tiene espacio para ser llamado en el rango de este mismo |

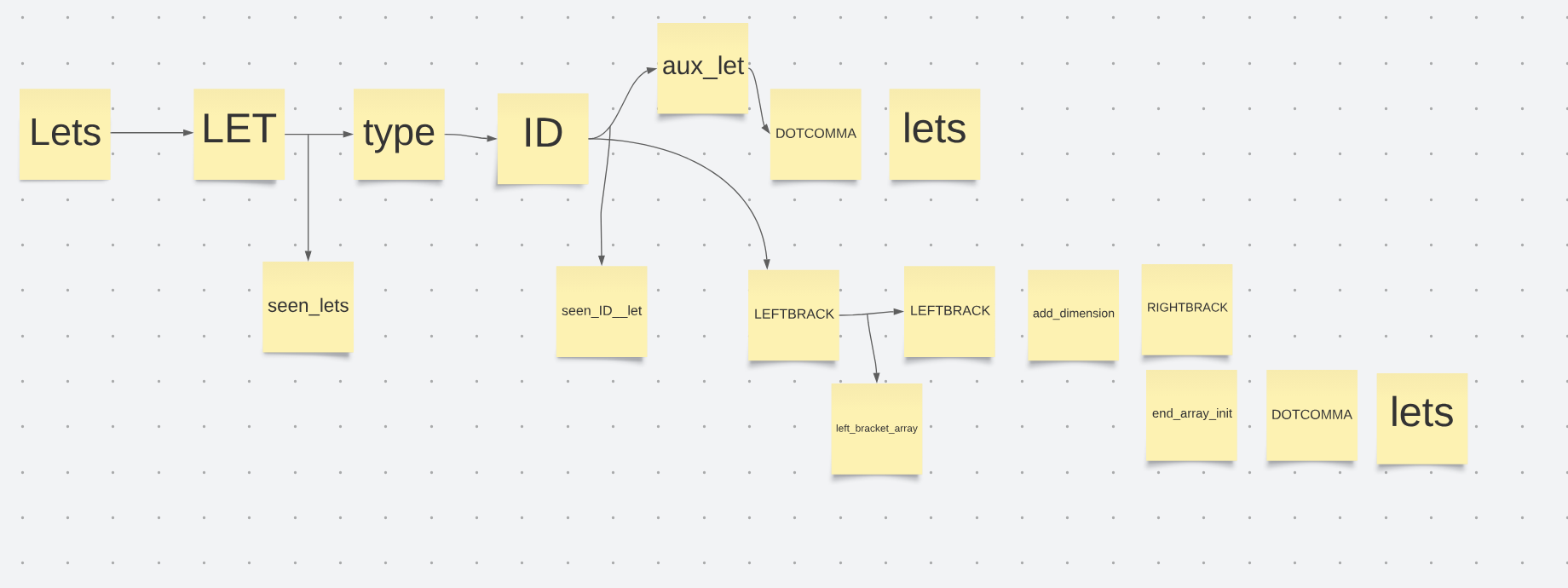
## 3.7 Diagramas de flujo

Diagrama de programa

Detecta cuando un programa inicia, guardando sus datos y pasando a las siguientes fases según estas van apareciendo.

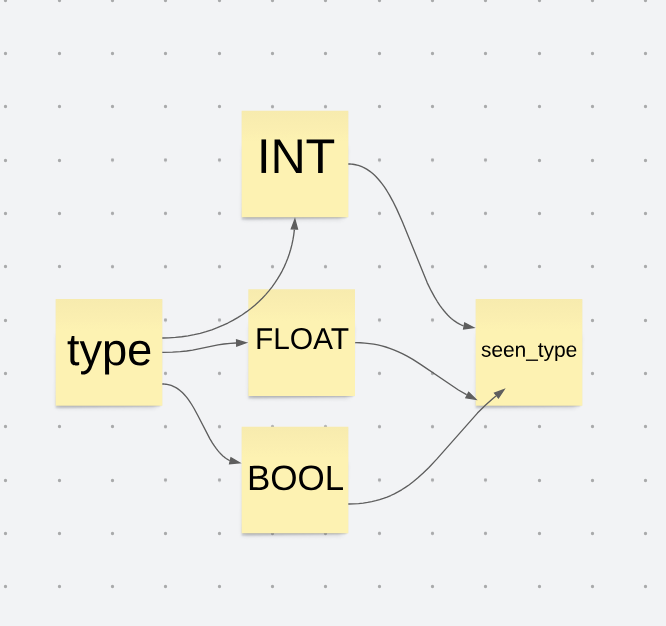


seen\_program guarda el nombre del programa, ademas de ponerlo como la funcion actual al no detectar otro



Lets detecta si lo que va a leer es un ID de variable o de array, toma su siguiente paso dependiendo esto, ya sea solo guardando el let en donde debe o creando los pasos necesarios para un array, se vuelve a llamar para buscar mas

seen\_lets solo guarda en la tabla, y la crea si esta no existe

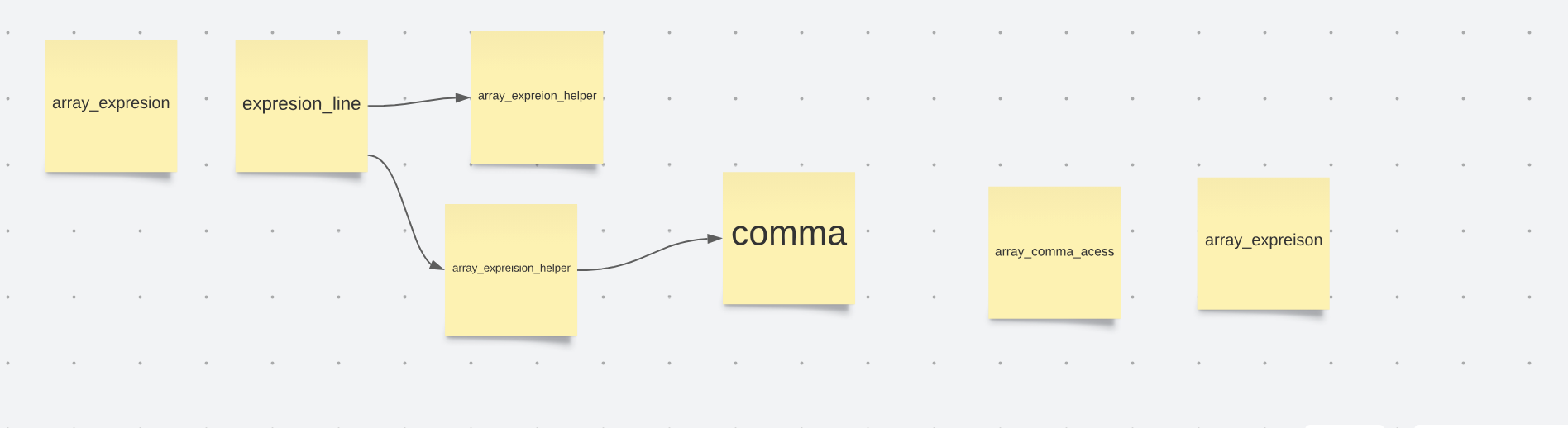


Type detecta de que tipo es la variable y la guarda en nuestra memoria

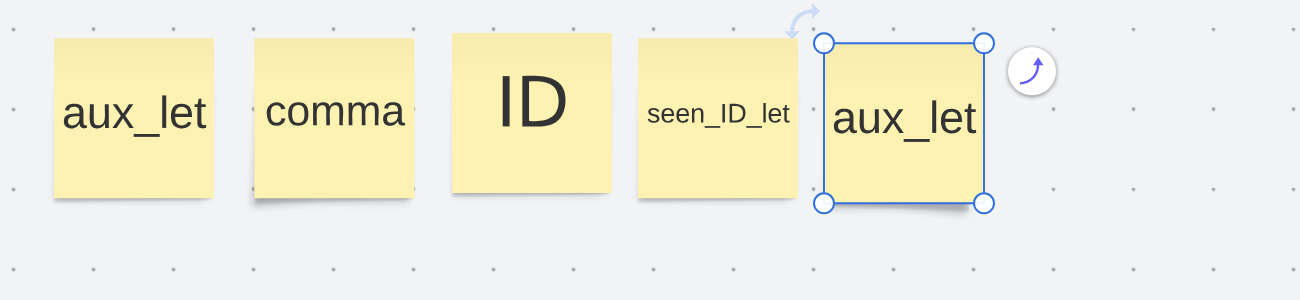


access\_array detecta el id y a donde quiere accesar, para despues buscar el valor en la tabla de variables

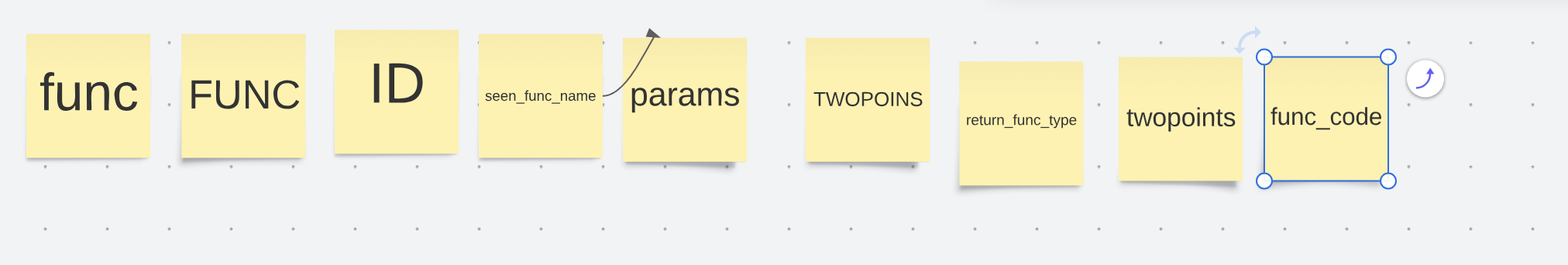
access\_array found guarda el valor de la variable de array’



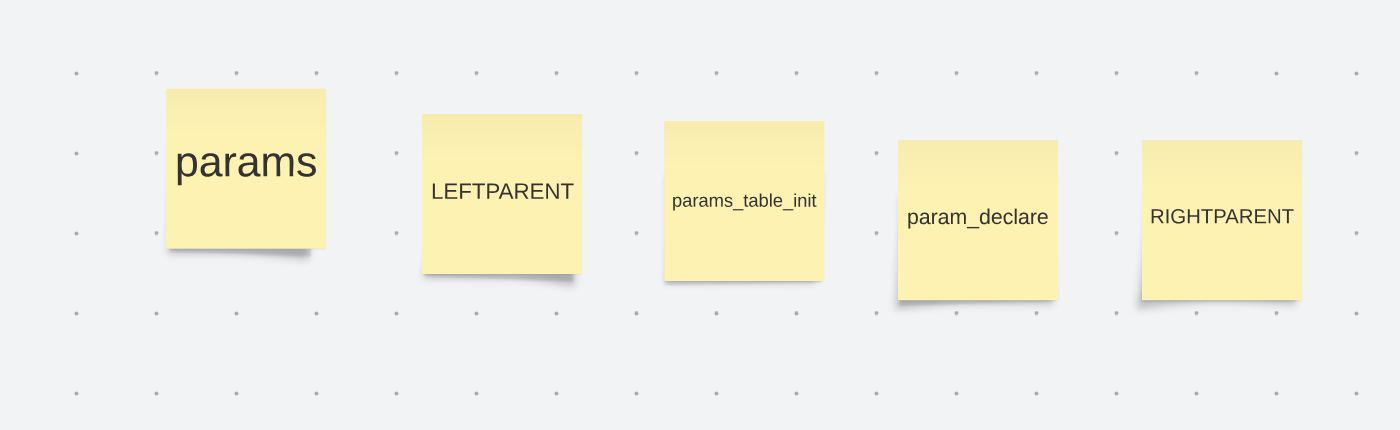
Array\_expresion detecta a que direccion quiere entrar, se puede llevar sobre si misma si necesita encontrar otra



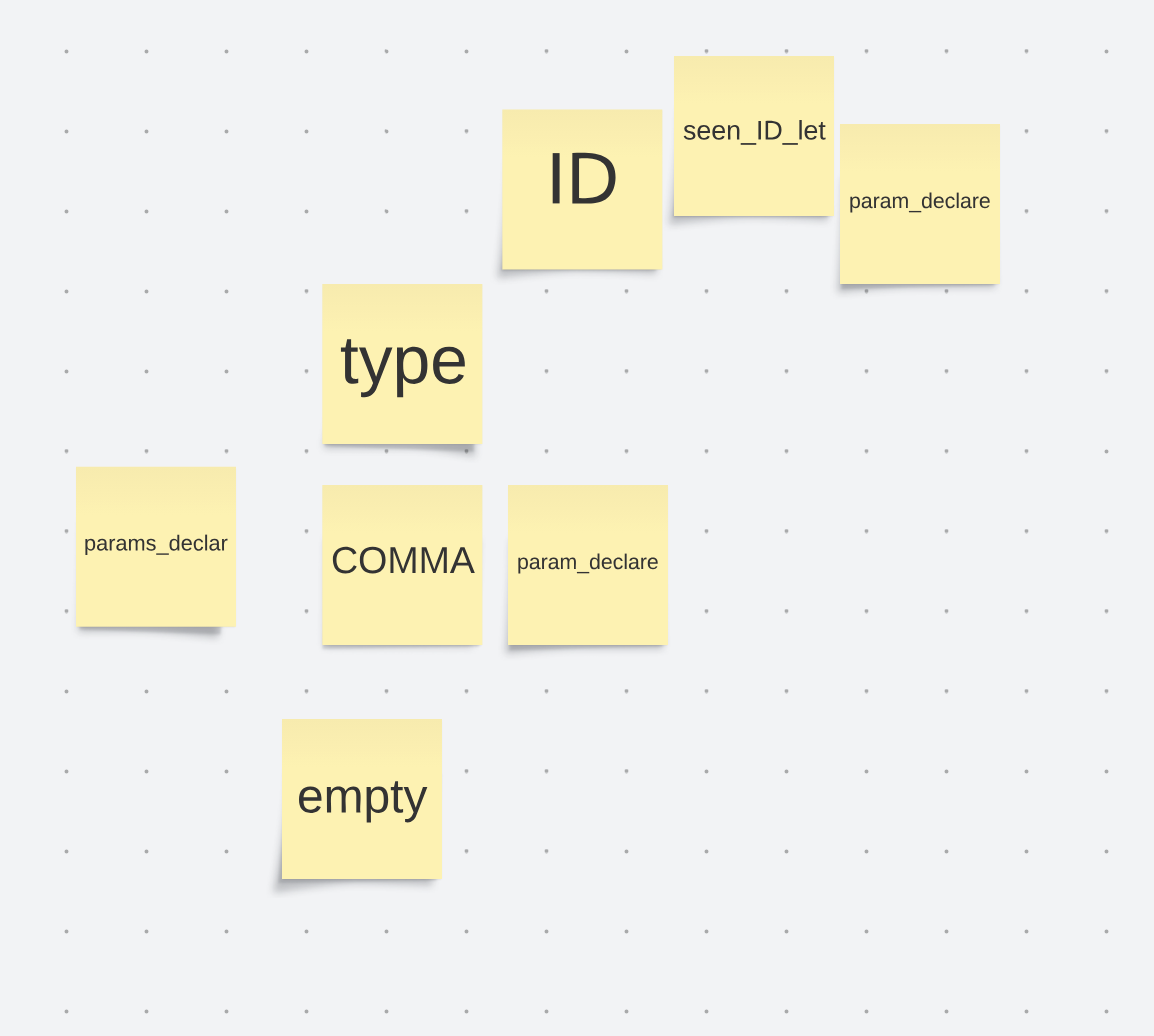
Aux let nos permite meter mas de un let identificando si hay coma y repitiendose a si mismo



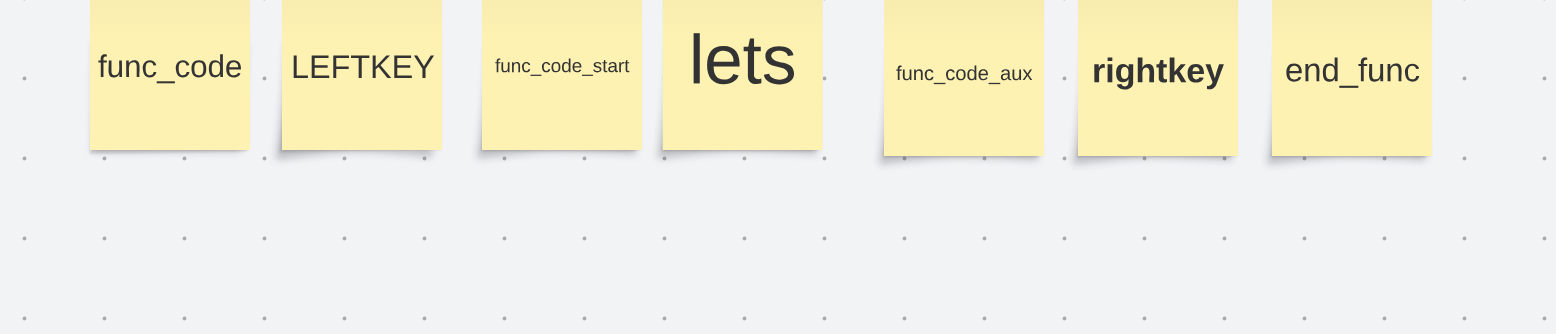
Al detectar un func, sabemos que una funcion va a ser creada, el seen\_func\_name guarda el nombre de la funcion, paramas se prepara para recibir parametros mientras que return\_func\_type solo guarda el typo de funcion, todo el codigo esta en func\_code



params checa si hay un ( para iniciar la tabla de parametro y comenzar a registrarlos



params\_declar puede ir a tres caminos si nada mas shinee id se guarda su tipo y se guarda en las variables , si es comma se repite si no se termina



func\_code checa si hay una key de esta forma sabe que hay codigo en la funcion para despues ver si hay variables que se declaren , si no va al codigo y a expresiones para luego avisar que esta vacio

## 3.8 Cubo Semántico

El cubo semántico nos sirve para hacer la validación si dos tipos que se encuentran en alguna operación, se verifica el operador aritmético o de comparación para saber si estos dos son compatibles.

cuboSemantico = {

'+': {

'int': {

'int': 'int',

'float': 'float',

'string': 'string',

'bool': 'int'

},

'float': {

'int': 'float',

'float': 'float',

'string': 'string',

'bool': 'float'

},

'string': {

'int': 'string',

'float': 'string',

'string': 'string',

'bool': 'string'

},

'bool': {

'int': 'int',

'float': 'float',

'string': 'string',

'bool': 'bool'

},

},

'-': {

'int': {

'int': 'int',

'float': 'float',

'string': 'error',

'bool': 'error'

},

'float': {

'int': 'float',

'float': 'float',

'string': 'error',

'bool': 'error'

},

'string': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

'bool': {

'int': 'int',

'float': 'float',

'string': 'error',

'bool': 'bool'

},

},

'\*': {

'int': {

'int': 'int',

'float': 'float',

'string': 'error',

'bool': 'int'

},

'float': {

'int': 'float',

'float': 'float',

'string': 'error',

'bool': 'float'

},

'string': {

'int': 'string',

'float': 'error',

'string': 'error',

'bool': 'string'

},

'bool': {

'int': 'int',

'float': 'float',

'string': 'string',

'bool': 'bool'

},

},

'/': {

'int': {

'int': 'float',

'float': 'float',

'string': 'error',

'bool': 'int'

},

'float': {

'int': 'float',

'float': 'float',

'string': 'error',

'bool': 'float'

},

'string': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

'bool': {

'int': 'float',

'float': 'float',

'string': 'int',

'bool': 'bool'

},

},

'^': {

'int': {

'int': 'int',

'float': 'float',

'string': 'error',

'bool': 'error'

},

'float': {

'int': 'float',

'float': 'float',

'string': 'error',

'bool': 'error'

},

'string': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

'bool': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

},

'>': {

'int': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'float': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'string': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

'bool': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

},

'<': {

'int': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'float': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'string': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

'bool': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

},

'<=': {

'int': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'float': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'string': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

'bool': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

},

'>': {

'int': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'float': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'string': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

'bool': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

},

'>=': {

'int': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'float': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'string': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

'bool': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

},

'!=': {

'int': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'float': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'string': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

'bool': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

},

'==': {

'int': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'float': {

'int': 'bool',

'float': 'bool',

'string': 'error',

'bool': 'error'

},

'string': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

'bool': {

'int': 'error',

'float': 'error',

'string': 'error',

'bool': 'error'

},

}

}



# 3.8 Memoria Virtual

La memoria virtual nos sirve a la hora de la comparación para tener un control de las cosas que necesitamos, desde saber qué saltos dar hasta tener un control del tamaño de ciertos requerimientos.

A continuación veremos el objeto de memoria virtual utilizando en Gallagher con una explicación sobre casa parte de este

| Objeto | Uso |
| --- | --- |
| progName | Espacio utilizado para guardar el nombre del programa |
| currentFunc | Espacio para saber cual es la función que utilizamos actulamente |
| currentType | Espacio para saber cual es el tipo del let utilizado actualmente |
| currentLet | Espacio para saber cual let utilizamos por ultima vez |
| currentArray | Espacio para saber cual Arrey utilizamos por última vez o en cual nos encontramos |
| currentSign | Espacio para saber cual símbolo vimos por ultima ves |
| currentFuncCalled | Espacio para saber cual funcion fue llamada la ultima vez |
| currentFuncQuadStart | Funcion para saber en que quad inicio la ultima funcion llamada |
| currentFuncCalledValue | Funcion que nos sirve para saber |
| quadCount | Contador para saber que quad es el ultimo en agregarse |
| tempCount | Contador para saber que temporal vas actulamente |
| paramCounter | Contador para saber qué parámetro vas actulamente |
| funciones | Directorio de funciones |
| quads | objeto para guardar todos los cuadrantes |
| operands | array para saber que operandos estan esperando ser asignados |
| operators | array para saber que operadores estan esperando ser asignados |
| types | array para saber que tipos estan esperando ser revisados |
| jumps | array para saber que saltos estan esperando ser asignados |
| funcCalls | las funciones que fueron llamadas |

Virtual Memory

global --

i : 0

fib : 1

main : 2

constant --

1 : 6001

2 : 6002

11 : 6003

0 : 6004

local --

fib : {'n': 2001, 'value': 2002, 'temp1': 4001, 'temp2': 4002, 'temp3': 4003, 'temp4': 4004, 'temp5': 4005, 'temp6': 4006}

main : {'temp1': 4001}

Execution Memory

{'global': {0: None, 1: 89, 2: 0}, 'local': [{4001: 89}], 'constant': {6001: 1, 6002: 2, 6003: 11, 6004: 0}, 'excectuionPointer': 26, 'localPointer': 0, 'endFuncPointer': []}

{'progName': 'fibonacci', 'currentFunc': 'main', 'currentType': 'int', 'currentLet': 'value', 'currentArray': '', 'currentSign': '=', 'currentFuncCalled': '', 'currentFuncQuadStart': 20, 'currentFuncCalledValue': '', 'quadCount': 26, 'tempCount': 2, 'paramCounter': 0, 'funciones': {'fibonacci': {'type': 'Programa', 'letsTable': {'i': {'type': 'int'}, 'fib': {'type': 'test', 'value': None}, 1: {'type': 'pruebaa'}, 2: {'type': 'pruebaa'}, 'main': {'type': 'test', 'value': None}, 11: {'type': 'pruebaa'}, 0: {'type': 'pruebaa'}}}, 'fib': {'type': 'int', 'quadStart': 2, 'letsTable': {'n': {'type': 'int'}, 'value': {'type': 'int'}}, 'paramsTable': {}}, 'main': {'type': 'int', 'quadStart': 20, 'letsTable': {}, 'paramsTable': {}}}, 'quads': [[], ['GOTO', '', '', 20], ['<=', 2001, 6001, 4001], ['GOTOF', 4001, '', 6], ['=', 2001, '', 2002], ['GOTO', '', '', 18], ['ERA', '', '', 'fib'], ['-', 2001, 6001, 4002], ['PARAMATER', 4002, '', 0], ['GOTOFUNC', 'fib', '', 2], ['=', 1, '', 4003], ['ERA', '', '', 'fib'], ['-', 2001, 6002, 4004], ['PARAMATER', 4004, '', 0], ['GOTOFUNC', 'fib', '', 2], ['=', 1, '', 4005], ['+', 4003, 4005, 4006], ['=', 4006, '', 2002], ['=', 2002, '', 1], ['ENDFUNC', '', '', ''], ['ERA', '', '', 'fib'], ['PARAMATER', 6003, '', 0], ['GOTOFUNC', 'fib', '', 2], ['=', 1, '', 4001], ['PRINTG', '', '', 4001], ['=', 6004, '', 2], ['ENDALL', '', '', '']], 'operators': ['('], 'operands': [], 'types': [], 'jumps': [], 'funcCalls': [], 'arrayDeclaration': False, 'dimensions': 1, 'arrayR': 1, 'dimensionStacks': [], 'functionCalled': 'fib'}



4 Máquina Virtual

La máquina virtual es lo que nos permite convertir los cuádruplos a acciones y valores que busca el usuario con su código, esta cuenta con su propia memoria para tener un control de lo que lleva.

La máquina virtual tiene su memoria y lee quad por quad asignando valores dependiendo de estos.

Sin la máquina no sucedería nada.

5 Pruebas

Fibonacci Recursivo

Programa fibonacci;

let int i;

func fib(int n): int :{

let int value;

if(n <= 1){

value = n;

}else{

value = @fib(n - 1) + @fib(n - 2);

}

return value;

}

func main(): int : {

printG(@fib(11));

return 0;

}



Virtual Memory

global --

i : 0

fib : 1

main : 2

constant --

1 : 6001

2 : 6002

11 : 6003

0 : 6004

local --

fib : {'n': 2001, 'value': 2002, 'temp1': 4001, 'temp2': 4002, 'temp3': 4003, 'temp4': 4004, 'temp5': 4005, 'temp6': 4006}

main : {'temp1': 4001}

Execution Memory

{'global': {0: None, 1: 89, 2: 0}, 'local': [{4001: 89}], 'constant': {6001: 1, 6002: 2, 6003: 11, 6004: 0}, 'excectuionPointer': 26, 'localPointer': 0, 'endFuncPointer': []}

{'progName': 'fibonacci', 'currentFunc': 'main', 'currentType': 'int', 'currentLet': 'value', 'currentArray': '', 'currentSign': '=', 'currentFuncCalled': '', 'currentFuncQuadStart': 20, 'currentFuncCalledValue': '', 'quadCount': 26, 'tempCount': 2, 'paramCounter': 0, 'funciones': {'fibonacci': {'type': 'Programa', 'letsTable': {'i': {'type': 'int'}, 'fib': {'type': 'test', 'value': None}, 1: {'type': 'pruebaa'}, 2: {'type': 'pruebaa'}, 'main': {'type': 'test', 'value': None}, 11: {'type': 'pruebaa'}, 0: {'type': 'pruebaa'}}}, 'fib': {'type': 'int', 'quadStart': 2, 'letsTable': {'n': {'type': 'int'}, 'value': {'type': 'int'}}, 'paramsTable': {}}, 'main': {'type': 'int', 'quadStart': 20, 'letsTable': {}, 'paramsTable': {}}}, 'quads': [[], ['GOTO', '', '', 20], ['<=', 2001, 6001, 4001], ['GOTOF', 4001, '', 6], ['=', 2001, '', 2002], ['GOTO', '', '', 18], ['ERA', '', '', 'fib'], ['-', 2001, 6001, 4002], ['PARAMATER', 4002, '', 0], ['GOTOFUNC', 'fib', '', 2], ['=', 1, '', 4003], ['ERA', '', '', 'fib'], ['-', 2001, 6002, 4004], ['PARAMATER', 4004, '', 0], ['GOTOFUNC', 'fib', '', 2], ['=', 1, '', 4005], ['+', 4003, 4005, 4006], ['=', 4006, '', 2002], ['=', 2002, '', 1], ['ENDFUNC', '', '', ''], ['ERA', '', '', 'fib'], ['PARAMATER', 6003, '', 0], ['GOTOFUNC', 'fib', '', 2], ['=', 1, '', 4001], ['PRINTG', '', '', 4001], ['=', 6004, '', 2], ['ENDALL', '', '', '']], 'operators': ['('], 'operands': [], 'types': [], 'jumps': [], 'funcCalls': [], 'arrayDeclaration': False, 'dimensions': 1, 'arrayR': 1, 'dimensionStacks': [], 'functionCalled': 'fib'}

Array 3 dimensiones, asignacion e impresion

Programa Arrays;

let int a;

func main(): int : {

let int b;

let int una [1 5, 1 4, 1 5];

b = 3;

una[2,4,4] = 1 + b;

una[1,4,5] = 2;

printG(una[2,4,4]);

printG(una[1,4,5]);

}



/usr/bin/python3 /Users/chuca/PycharmProjects/compiladorSL/main.py

Hi Welcome to Gallagher compiler

To start enter the path to your text file with your Gallagher code to start

#Pss rembmer to enter the full path, like this : /Users/chuca/PycharmProjects/compiladorSL/gramatica/testsFiles/arrays.txt

Your file path: /Users/chuca/PycharmProjects/compiladorSL/gramatica/testsFiles/arrays.txt

4

2

CUADRUPLOS

SIGN | LEFT | RIGHT | RESULT

[[], ['GOTO', '', '', 2], ['=', 6001, '', 2001], ['VERIFY', 6002, 6003, 6004], ['\*', 6002, 6005, 4001], ['VERIFY', 6006, 6003, 6006], ['\*', 6006, 6003, 4002], ['VERIFY', 6006, 6003, 6004], ['+', 6006, 4002, 4003], ['+', 4003, 6007, 4004], ['+', 4004, 6008, 4005], ['+', 6003, 2001, 4006], ['=', 4006, '', '\*4005'], ['VERIFY', 6003, 6003, 6004], ['\*', 6003, 6005, 4007], ['VERIFY', 6006, 6003, 6006], ['\*', 6006, 6003, 4008], ['VERIFY', 6004, 6003, 6004], ['+', 6004, 4008, 4009], ['+', 4009, 6007, 4010], ['+', 4010, 6008, 4011], ['=', 6002, '', '\*4011'], ['VERIFY', 6002, 6003, 6004], ['\*', 6002, 6005, 4012], ['VERIFY', 6006, 6003, 6006], ['\*', 6006, 6003, 4013], ['VERIFY', 6006, 6003, 6004], ['+', 6006, 4013, 4014], ['+', 4014, 6007, 4015], ['+', 4015, 6008, 4016], ['PRINTG', '', '', '\*4016'], ['VERIFY', 6003, 6003, 6004], ['\*', 6003, 6005, 4017], ['VERIFY', 6006, 6003, 6006], ['\*', 6006, 6003, 4018], ['VERIFY', 6004, 6003, 6004], ['+', 6004, 4018, 4019], ['+', 4019, 6007, 4020], ['+', 4020, 6008, 4021], ['PRINTG', '', '', '\*4021'], ['ENDALL', '', '', '']]

Virtual Memory

global --

a : 0

main : 1

constant --

3 : 6001

2 : 6002

1 : 6003

5 : 6004

20 : 6005

4 : 6006

26 : 6007

2002 : 6008

local --

main : {'b': 2001, 'una': 2002, '': [2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101], 'temp1': 4001, 'temp2': 4002, 'temp3': 4003, 'temp4': 4004, 'temp5': 4005, 'temp6': 4006, 'temp7': 4007, 'temp8': 4008, 'temp9': 4009, 'temp10': 4010, 'temp11': 4011, 'temp12': 4012, 'temp13': 4013, 'temp14': 4014, 'temp15': 4015, 'temp16': 4016, 'temp17': 4017, 'temp18': 4018, 'temp19': 4019, 'temp20': 4020, 'temp21': 4021}

Execution Memory

{'global': {0: None, 1: None}, 'local': [{2001: 3, 2002: None, 4001: 40, 4002: 4, 4003: 8, 4004: 34, 4005: 4, 4006: 4, 4007: 20, 4008: 4, 4009: 9, 4010: 35, 4011: 2, 4012: 40, 4013: 4, 4014: 8, 4015: 34, 4016: 2036, 4017: 20, 4018: 4, 4019: 9, 4020: 35, 4021: 2037, 4: None, 2036: 4, 2: None, 2037: 2}], 'constant': {6001: 3, 6002: 2, 6003: 1, 6004: 5, 6005: 20, 6006: 4, 6007: 26, 6008: 2002}, 'excectuionPointer': 40, 'localPointer': 0, 'endFuncPointer': []}

{'progName': 'Arrays', 'currentFunc': 'main', 'currentType': 'int', 'currentLet': 'b', 'currentArray': 'una', 'currentSign': '=', 'currentFuncCalled': '', 'currentFuncQuadStart': 2, 'currentFuncCalledValue': '', 'quadCount': 40, 'tempCount': 22, 'paramCounter': 0, 'funciones': {'Arrays': {'type': 'Programa', 'letsTable': {'a': {'type': 'int'}, 'main': {'type': 'test', 'value': None}, 3: {'type': 'pruebaa'}, 2: {'type': 'pruebaa'}, 4: {'type': 'pruebaa'}, 1: {'type': 'pruebaa'}, 5: {'type': 'pruebaa'}}}, 'main': {'type': 'int', 'quadStart': 2, 'letsTable': {'b': {'type': 'int'}, 'una': {'type': 'int', 'dimensionsNodes': [{'arrayStart': 1, 'arrayEnd': 5, 'm': 20.0}, {'arrayStart': 1, 'arrayEnd': 4, 'm': 1.0}, {'arrayStart': 1, 'arrayEnd': 5, 'm': 5.0, 'k': 26.0, 'end': True}]}}, 'paramsTable': {}}}, 'quads': [[], ['GOTO', '', '', 2], ['=', 6001, '', 2001], ['VERIFY', 6002, 6003, 6004], ['\*', 6002, 6005, 4001], ['VERIFY', 6006, 6003, 6006], ['\*', 6006, 6003, 4002], ['VERIFY', 6006, 6003, 6004], ['+', 6006, 4002, 4003], ['+', 4003, 6007, 4004], ['+', 4004, 6008, 4005], ['+', 6003, 2001, 4006], ['=', 4006, '', '\*4005'], ['VERIFY', 6003, 6003, 6004], ['\*', 6003, 6005, 4007], ['VERIFY', 6006, 6003, 6006], ['\*', 6006, 6003, 4008], ['VERIFY', 6004, 6003, 6004], ['+', 6004, 4008, 4009], ['+', 4009, 6007, 4010], ['+', 4010, 6008, 4011], ['=', 6002, '', '\*4011'], ['VERIFY', 6002, 6003, 6004], ['\*', 6002, 6005, 4012], ['VERIFY', 6006, 6003, 6006], ['\*', 6006, 6003, 4013], ['VERIFY', 6006, 6003, 6004], ['+', 6006, 4013, 4014], ['+', 4014, 6007, 4015], ['+', 4015, 6008, 4016], ['PRINTG', '', '', '\*4016'], ['VERIFY', 6003, 6003, 6004], ['\*', 6003, 6005, 4017], ['VERIFY', 6006, 6003, 6006], ['\*', 6006, 6003, 4018], ['VERIFY', 6004, 6003, 6004], ['+', 6004, 4018, 4019], ['+', 4019, 6007, 4020], ['+', 4020, 6008, 4021], ['PRINTG', '', '', '\*4021'], ['ENDALL', '', '', '']], 'operators': [], 'operands': ['temp1', 'temp4', 'temp7', 'temp10', 'temp12', 'temp15', 'temp17', 'temp20'], 'types': [], 'jumps': [], 'funcCalls': [], 'arrayDeclaration': True, 'dimensions': 1, 'arrayR': 1.0, 'dimensionStacks': []}

Process finished with exit code 0



6 Video

<https://youtu.be/ttib7xK092A>

LINK a github

https://github.com/diegoGPD/GallagherComp