Tecnológico de Monterrey

Documentación del Proyecto

Miguel Ángel Tornero Carrillo

A00820449

Diseño de compiladores (Gpo 2)

Ing. Elda G. Quiroga

Dr. Héctor Ceballos

5 de junio de 2023

Descripción del proyecto

Propósito y Alcance del Proyecto

El propósito del proyecto fue aprender las bases requeridas para la elaboración de un compilador, la cual incluye la creación y el diseño de un lenguaje de programación. El lenguaje es un lenguaje simple procedural que no utiliza cosas como memoria dinámica, polimorfismo, tipos dinámicos, scopes anidados, etc.. La sintaxis es similar al lenguaje C y cuenta con dos partes: un compilador que compila un archivo de texto a un archivo objeto, y una máquina virtual que ejecuta el archivo objeto.

Requerimientos

Los requerimientos del lenguaje de programación mínimos fueron los siguientes:

- Variables locales y globales
- Variables dimensionadas (arreglos y matrices)
- Funciones void y funciones que regresan un valor
- Soporte de recursividad
- Estatutos no lineales IF-ELSE
- Estatutos cíclicos WHILE y FOR
- Poder imprimir valores y letreros en consola
- Poder leer una entrada del usuario
- Alguna funcionalidad extra proveída por el lenguaje

Proceso

El proceso para este proyecto se constituyó principalmente en hacer grandes avances durante unas semanas, mientras que en otras se descansó para enfocarse en otras materias, teniendo en cuenta que el proyecto se mantuviera al tanto. Debido a que se hizo de manera individual, se pudo trabajar a ritmo propio. Para un reporte más detallado de la frecuencia del trabajo hecho, se puede consultar los commits en el repositorio de GitHub (https://github.com/A00820449/antlr4-final-project/commits/main).

El proceso me ayudó a entender de manera más fundamental cómo funcionan los lenguajes de programación. El poner en práctica los conocimientos vistos realmente clarificó conceptos que se vieron previamente en mi carrera, como el análisis léxico-sintáctico, errores de semántica estáticos y dinámicos, la diferencia de memoria en Heap y en Stack, entre muchas otras cosas.

Descripción del lenguaje

El lenguaje Image Manipulation Language (IML) tiene como propósito facilitar la manipulación básica de imágenes. Las funcionalidades básicas incluyen recortar una imágen (crop), cambiar

tamaño (resize), rotar la imagen (rotate), y voltear la imagen (flip). Para hacer estas operaciones simplemente se tiene que cargar una imagen con la función *load()* y al final puedes guardar el búfer modificado con la función *save()*.

Errores

Existen varios errores que pueden ocurrir durante la compilación y la ejecución del programa.

Los errores que pueden pasar durante la compilación (errores estáticos) incluyen:

- Errores de sintaxis (autogenerados por ANTLR)
- Error de tipos, puede incluir tanto operaciones no soportadas, como una discordancia en el tipo de dato esperado (Type Mismatch)
- Error de valores de dimensiones no positivos (Dimensions must be positive)
- Error de duplicación de ID tanto para variables como funciones (Duplicate ID)
- Error de dimensiones para arreglos y matrices (Missing dimensions, too many dimensions)
- Error al encontrar una variable o función no declarada (variable not defined, function not defined)
- Error de uso de estatuto break afuera de un ciclo
- Error de uso de una función void en una expresión.
- Error de número de argumentos en una llamada de función (too many arguments, not enough arguments)

Los errores que pueden pasar durante ejecución (errores dinámicos) incluyen:

- Error de division por zero
- Error de índice fuera de rango (Index out of bounds)
- Error de acceso a una variable no inicializada
- Error de número fuera de rango para las funciones trigonométricas asin, acos, y atan.
- Error de entrada no numérica por parte del usuario
- Error de lectura del archivo de imagen
- Error al guardar la imagen
- Error al intentar hacer una operación de imágen sin primero cargar una imágen con *load()*.
- Error de Heap out of memory de JavaScript
- Errores no previstos de JavaScript

Cabe mencionar que no hay un límite puesto para las llamadas recursivas, por lo que una llamada recursiva puede teóricamente ejecutarse una infinidad de veces, con el único límite siendo la cantidad de memoria alocada para el heap del programa.

Descripción del compilador

Para este proyecto, se decidió utilizar la herramienta ANTLRv4 (https://www.antlr.org/). ANTLR puede generar archivos para un analizador léxico-sintáctico para diferentes lenguajes de programación. Para este proyecto se utilizó el lenguaje JavaScript. Para poder hacer uso de los archivos de JS generados, se decidió utilizar la herramienta NodeJS (https://nodejs.org/) la cual permite ejecutar código de JS en una línea de comandos, sin necesidad de un navegador de web. NodeJS tiene un administrador de dependencias llamado NPM (https://www.npmjs.com/), la cual permite la distribución de las dependencias necesarias para ejecutar el compilador. Para este proyecto, principalmente se utilizó el sistema operativo Linux (distribución Fedora 38: https://fedoraproject.org/) como ambiente de desarrollo, pero teóricamente puede ser ejecutado en cualquier ambiente que soporte la versión más actual de NodeJS y ANTLR.

Análisis léxico

Patrones de construcción

El lenguaje cuenta con 5 token los cuales utilizan un patrón de construcción, a continuación se describen los patrones utilizando expresiones regulares:

Identificadores para variables y funciones (ID)	`[_a-zA-Z][_a-zA-Z0-9]*`
Constante numérica (NUM_CTE)	`[0-9]+(\.[0-9]+([Ee][-+]?[0-9]+)?)?`
Letrero (STR_CTE)	`"([^"\\\n]|\\[^\n])*"`
Comentario, el cual es ignorado (COMMENT)	`#[^\n]*\n`
Espacio en blanco, el cual es ignorado	`[ \n\r\t]+`

Enumeración

Incluyendo estos 5 tokens, existen un total de 68 tokens en el lenguaje, el cual incluye palabras reservadas. A continuación se muestra la enumeración final de estos tokens:

`'program' = 1`	`'%' = 35`
`':' = 2`	`'!' = 36`
`'var' = 3`	`'true' = 37`
`',' = 4`	`'false' = 38`

';' = 5	'PI' = 39
'number' = 6	'print' = 40
'boolean' = 7	'load' = 41
'[' = 8	'save' = 42
']' = 9	'read' = 43
'-' = 10	'main' = 44
'function' = 11	'trunc' = 45
'(' = 12	'round' = 46
')' = 13	'floor' = 47
'void' = 14	'ceil' = 48
'{' = 15	'isInteger' = 49
'}' = 16	'pow' = 50
'=' = 17	'sin' = 51
'return' = 18	'cos' = 52
'while' = 19	'tan' = 53
'for' = 20	'asin' = 54
'break' = 21	'acos' = 55
'if' = 22	'rand' = 56
'else' = 23	'getHeight' = 57
'&' = 24	'getWidth' = 58
'|' = 25	'crop' = 59
'==' = 26	'resize' = 60
'!=' = 27	'rotate' = 61
'>' = 28	'flipVertically' = 62
'>=' = 29	'flipHorizontally' = 63
'<' = 30	ID = 64
<=' = 31	NUM_CTE = 65

'+' = 32	STR_CTE = 66
'*' = 33	COMMENT = 67
'/' = 34	WS = 68

Análisis sintáctico

Para describir la gramática formal, se usará la notación Augmented Backus-Naur Form (ABNF), como está descrita en RFC 5234 (https://www.rfc-editor.org/info/rfc5234), con los tokens descritos en la sección anterior:

```
program = "program" ID ":" *var_decl *fun_decl "main" ":" block

vars = "var" var_type ID *("," ID) ";"

var_type = basic_type *2("[" expression "]")

basic_type = "number" / "boolean"

fun_decl = "function" ("void" / basic_type) ID params ":"
           *var_decl block

params = "(" *1(basic_type ID *(basic_type ID)) ")"

block = "{" *statement "}"

statement = assignment_stmt / print_stmt / read_stmt
          / load_stmt / save_stmt / if_else_stmt
          / while_stmt / for_stmt / break_stmt
          / fun_call_stmt / return_stmt / resize_stmt
          / crop_stmt / rotate_stmt / flip_v_stmt / flip_h_stmt

return_stmt = "return" *1expression ";"

break_stmt = "break" ";"

print_stmt = "print" "(" *1(printable *("," printable)) ")" ";"
printable = expression / STR_CTE

read_stmt = "read" "(" STR_CTE ")" ";"

load_stmt = "load" "(" STR_CTE ")" ";"

save_stmt = "save" "(" STR_CTE ")" ";"

resize_stmt = "resize" "(" expression "," expression ")" ";"

crop_stmt = "crop" "(" expression "," expression
```

```
            "," expression "," expression ")" ";"

rotate_stmt = "rotate" "(" expression ")" ";"

flip_v_stmt = "flipVertically" "(" ")" ";"

flip_h_stmt = "flipHorizontally" "(" ")" ";"

fun_call_stmt = fun_call ";"

if_else_stmt = "if" "(" expression ")" block *1("else" block)

while_stmt = "while" "(" expression ")" block

for_stmt = "for" "(" *1assignment ";" expression ";"
           *1assignment ")" block

assignment_stmt = assignment ";"

var_access = non_dim_access / arr_access / mat_access

non_dim_access = ID

arr_access = ID "[" expression "]"

mat_access = ID "[" expression "]" "[" expression "]"

assignment = var_access "=" expression ";"

expression = conjunction *("&" conjunction)

conjunction = relation *("|" relation)

relation = addition *(("==" / "!=" / ">" / ">=" / "<" / "<=")
           addition)

addition = term *(("+" / "-") term)

term = factor *(("*" / "/" / "%") factor)

factor = *1("!" / "-") negation

negation = paren_exp / atom

paren_exp = "(" expression ")"

atom = var_access / fun_call / literal

literal = NUM_CTE / "false" / "true" / "PI"

fun_call = round_fun_call / trunc_fun_call / floor_fun_call
```

```
          / ceil_fun_call / is_integer_fun_call / pow_fun_call
          / sin_fun_call / cos_fun_call / tan_fun_call
          / asin_fun_call / acos_fun_call / atan_fun_call
          / height_fun_call / width_fun_call / rand_fun_call
          / custom_fun_call

round_fun_call = "round" "(" expression ")"

trunc_fun_call = "trunc" "(" expression ")"

floor_fun_call = "floor" "(" expression ")"

ceil_fun_call = "ceil" "(" expression ")"

is_integer_fun_call = "isInteger" "(" expression ")"

pow_fun_call = "pow" "(" expression "," expression ")"

sin_fun_call = "sin" "(" expression ")"

cos_fun_call = "cos" "(" expression ")"

tan_fun_call = "tan" "(" expression ")"

asin_fun_call = "asin" "(" expression ")"

acos_fun_call = "acos" "(" expression ")"

atan_fun_call = "atan" "(" expression ")"

height_fun_call = "getHeight" "(" ")"

width_fun_call = "getWidth" "(" ")"

rand_fun_call = "rand" "(" ")"

custom_fun_call = ID "(" *1(expression *("," expression)) ")"
```

Para más información sobre esta notación, se puede consultar el siguiente artículo: https://en.wikipedia.org/wiki/Augmented_Backus%E2%80%93Naur_form

Generación de Código Intermedio y Análisis Semántico

Debido a que JavaScript usa cadenas de caracteres (String) como tipo de dato primitivo (https://developer.mozilla.org/en-US/docs/Glossary/Primitive), se decidó trabajar las direcciones y los códigos de ejecución como strings para facilitar el desarrollo del proyecto. Por esto, los cuádruplos que se generan son arreglos de cuatro elementos string, o nulos (null). Estos cuádruples son a su vez guardados en otro arreglo el cual es convertido en texto plano y guardado en un archivo usando la notación JSON, la cual tiene soporte nativo en JS (https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON). Este enfoque

aumentó la cantidad de memoria utilizada, pero también disminuyó considerablemente el tiempo de desarrollo.

Los cuádruplos generados quedaron en el siguiente formato:

```
[ OPERACIÓN, OP1, OP2, OP2, OP3 ]
```

Códigos de operacion

Los códigos de operación fueron los siguientes:

`ADD`	Suma los valores en OP1 y OP2, y guarda el resultado en OP3
`SUB`	Resta los valores en OP1 y OP2, y guarda el resultado en OP3
`MUL`	Multiplica los valores en OP1 y OP2, y guarda el resultado en OP3
`DIV`	Divide los valores en OP1 y OP2, y guarda el resultado en OP3. Este estatuto falla si OP2 es igual a cero.
`MOD`	Calcula el residuo de la división entre los valores en OP1 y OP2 y guarda el resultado en OP3
`EQ`	Calcula si el valor en OP1 es igual al valor en OP2, y guarda el resultado en OP3
`NE`	Calcula si el valor en OP1 es diferente al valor en OP2, y guarda el resultado en OP3
`GT`	Calcula si el valor en OP1 es mayor que el valor en OP2, y guarda el resultado en OP3
`GE`	Calcula si el valor en OP1 es mayor o igual al valor en OP2, y guarda el resultado en OP3
`LT`	Calcula si el valor en OP1 es menor que el valor en OP2, y guarda el resultado en OP3
`LE`	Calcula si el valor en OP1 es menor o igual al valor en OP2, y guarda el resultado en OP3
`AND`	Calcula la operación binaria AND entre los valores en OP1 y OP2, y guarda el resultado en OP3
`OR`	Calcula la operación binaria OR entre los valores en OP1 y OP2, y guarda el resultado en OP3
`NOT`	Calcula la negación binario de OP1, y guarda el resultado en OP3
`NEG`	Niega OP1 y guarda el resultado en OP3
`ASS`	Asigna el valor en OP1 a OP3

PRNT	Imprime el valor en OP1 en la consola
READ	Obtiene input del usuario desde la consola, lo interpreta como un número y lo guarda en OP3. Si la entrada dada no es un valor numérico válido, el estatuto falla
GOTO	Mueve el apuntador de ejecución al cuádruplo indicado en OP3
GOTF	Mueve el apuntador de ejecución al cuádruplo indicado en OP3 si OP1 tiene un valor falso
GOTT	Mueve el apuntador de ejecución al cuádruplo indicado en OP3 si OP1 tiene un valor verdadero
CALL	Inicializa una nueva memoria local, la coloca encima de la pila de ejecución, guarda el apuntador en la pila de apuntadores, y mueve el apuntador de ejecución al cuádruplo indicado en OP3
RTRN	Se deshace de la memoria en la cima de la pila de ejecución, obtiene el apuntador de la cima de la pila de apuntadores y restaura el apuntador actual a ese valor + 1
END	Termina la ejecución del programa
ADDP	Calcula la suma de la dirección OP1 y el valor en OP2, y guarda el resultado en OP3 (apuntador)
RANG	Calcula si el valor en OP1 es mayor o igual al valor en OP2, y menor que el valor en OP3, y de no ser así falla el programa.
TRUN	Trunca el valor en OP1 y guarda el resultado en OP3
ROUN	Redondea el valor en OP1 y guarda el resultado en OP3
FLOO	Calcula la operación piso con el valor en OP1 y guarda el resultado en OP3
CEIL	Calcula la operación techo con el valor en OP1 y guarda el resultado en OP3
ISIN	Calcula si el valor en OP1 es un valor entero o no, y guarda el resultado en OP3
POW	Eleva el valor en OP1 a la potencia del valor en OP2, y guarda el resultado en OP3
SIN	Calcula el seno del valor en OP1 y guarda el valor en OP3
COS	Calcula el coseno del valor en OP1 y guarda el valor en OP3
TAN	Calcula la tangente del valor en OP1 y guarda el valor en OP3
ASIN	Calcula el seno inverso del valor en OP1 y guarda el valor en OP3. Falla si el valor en OP1 es menor a 0 o mayor a 1.
ACON	Calcula el coseno inverso del valor en OP1 y guarda el valor en OP3. Falla si el valor en OP1 es menor a 0 o mayor a 1.

ATAN	Calcula la tangente inversa del valor en OP1 y guarda el valor en OP3. Falla si el valor en OP1 es menor a 0 o mayor a 1.
RAND	Genera un número aleatorio entre 0 y 1, y lo guarda en OP3
LOAD	Carga un archivo que contiene una imágen a memoria basándose en una cadena de caracteres guardada en OP1. Este estatuto puede fallar con un error de File System.
SAVE	Guarda el búfer modificado de la imagen en un archivo en disco bajo el nombre descrito en una cadena de caracteres guardada en OP1. Este estatuto puede fallar con un error de File System.
WIDT	Obtiene la anchura de la imágen cargada y guarda el valor en OP3. Falla si no hay una imágen cargada.
HEIG	Obtiene la altura de la imágen cargada y guarda el valor en OP3. Falla si no hay una imágen cargada.
SETX	Obtiene el valor en OP1 y lo guarda en memoria para usarlo en una operación de imágen (actualmente sólo para el valor X de crop(X,Y,W,H)). Falla si no hay una imágen cargada.
SETY	Obtiene el valor en OP1 y lo guarda en memoria para usarlo en una operación de imágen (actualmente sólo para el valor Y de crop(X,Y,W,H)). Falla si no hay una imágen cargada.
SETW	Obtiene el valor en OP1 y lo guarda en memoria para usarlo en una operación de imágen (actualmente sólo para el valor W de crop(X,Y,W,H) y resize(W,H)). Falla si no hay una imágen cargada.
SETH	Obtiene el valor en OP1 y lo guarda en memoria para usarlo en una operación de imágen (actualmente sólo para el valor H de crop(X,Y,W,H) y resize(W,H)). Falla si no hay una imágen cargada.
SETD	Obtiene el valor en OP1 y lo guarda en memoria para usarlo en una operación de imágen (actualmente sólo para el valor DEG de rotate(DEG)). Falla si no hay una imágen cargada.
CROP	Ejecuta la operación crop(X,Y,W,H) en la imagen en memoria con los valores actuales de X, Y, W, y H. Falla si no hay una imágen cargada.
RESZ	Ejecuta la operación resize(W,H) en la imágen en memoria con los valores actuales de W y H. Falla si no hay una imágen cargada.
ROTA	Ejecuta la operación rotate(DEG) en la imágen en memoria con el calor actual de DEG. Falla si no hay una imágen cargada.
FLIH	Ejecuta la operación flipHorizontally() en la imágen en memoria. Falla si no hay una imágen cargada.
FLIV	Ejecuta la operación flipVertically() en la imágen en memoria. Falla si no hay una

	imágen cargada.

Direcciones virtuales

Debido a que se utilizó strings para representar direcciones de memoria, se pudo simplificar el proceso poniendo prefijos que indican si una dirección es global, local, temporal, un apuntador, etc.

El formato para las direcciones fue el siguiente: `${prefijo}_{número}`

Por ejemplo, para la primera variable global, esta sería la dirección (tomando en cuenta que se empezó desde 0): `$g_0`. A su vez, esta sería la dirección de la quinta variable local: `$l_4`. La lista de todos los prefijos se describe a continuación:

`g`	Variable declarada global
`l`	Variable declarada local
`c`	Constante
`t`	Variable temporal global
`lt`	Variable temporal local
`$`	Variable apuntador temporal global
`l$`	Variable apuntador temporal local
`a`	Variable global usada para pasar argumentos a una función

Aparte de estos, también existen direcciones reservadas para usos específicos:

`$r`	Dirección global que guarda el valor que regresa una función
`$c_n`	Dirección que guarda el valor de carácter *newline* ('\n').
`$c_0`	Dirección que guarda el valor numérico 0.
`$c_pi`	Dirección que guarda el valor numérico de pi.
`$c_t`	Dirección que guarda el valor booleano verdadero
`$c_f`	Dirección que guarda el valor booleano falso

Diagramas de Sintaxis

(Nota: cada que se toma un operando de la pila de operandos, se libera si es una variable temporal. El contador de cuádruplos es aumentado por 1 cada vez que se genera uno.)

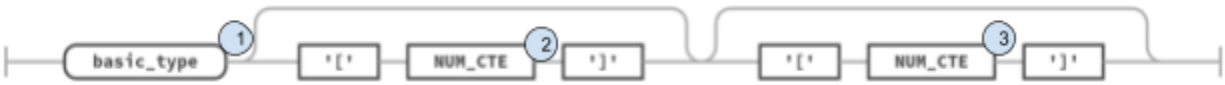
Programa (inicio)

1: Inicializar las estructuras de datos. Generar el cuádruplo GOTO y poner el contador en la pila de saltos

2: Llenar el cuádruplo GOTO generado al inicio.

1, 2: Cambiar el scope actual a '$global'

var_type

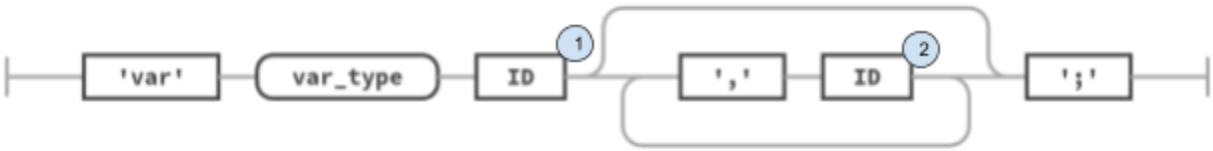


1: Actualizar el tipo de variable actual. Resetear dimensiones.

2: Agregar dimensión de variable actual (arreglo)

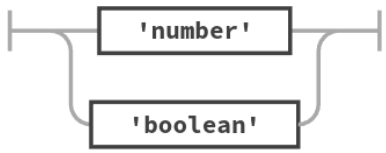
3: Agregar dimensión de variable actual (matriz)

var_decl



1, 2: Obtener dirección basada en el contador de variables. Dar de alta la variable usando la información tipo actual (tipo básico, dimensiones) en la tabla de variables globales si el scope es igual a '$global' o en la tabla de variables locales si no es así. Checar que no exista todavía antes de dar de alta. Aumentar el contador de variables, ya sea el local o el global, por 1 si no fue dimensionada, por el valor de la dimensión 1 si fue arreglo, o por el producto de la dimensión 1 por la dimensión 2 si fue matriz.

basic_type



fun_decl

1, 2: Actualizar el tipo de función actual con el tipo dado. Resetear la tabla de variables locales. Resetear contador de variables locales.

3: Dar de alta la función en la tabla de funciones usando el tipo de función actual y el contador de cuádruplo actual. Checa que no haya un duplicado. También cambia el scope actual al nombre de la función.

4: Genera el cuádruplo con operación ASS que asigna el valor default a $r (sólo si el tipo de función no es void). Genera el cuádruplo con la operación RTRN.

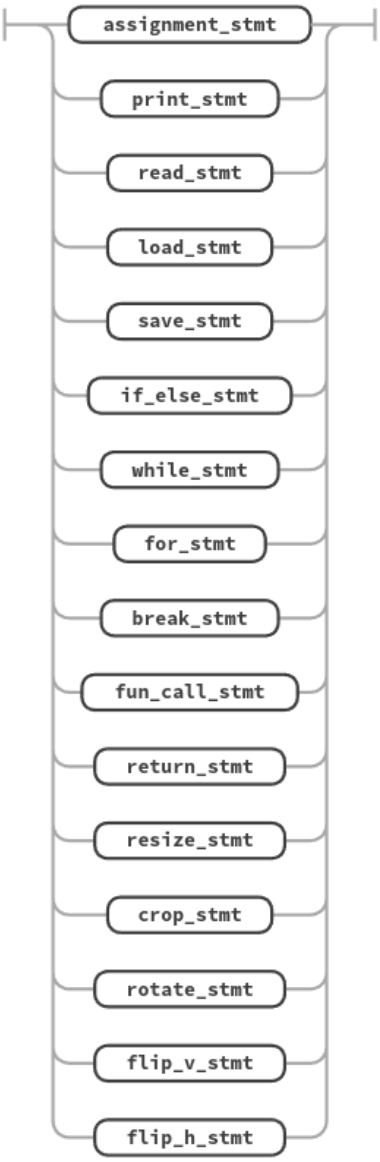
params

1, 3: Actualizar el tipo de variable actual. Resetear dimensiones. Agregar el tipo a la lista de parámetros en la tabla de funciones, utilizando el scope actual.

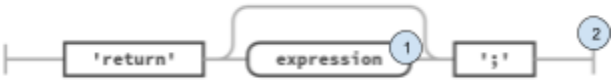
2, 4: Usar el contador de variables locales para obtener la dirección de la variable y darla de alta en la tabla de constantes, utilizando el tipo actual. Checar que no haya duplicados. Generar un cuádruplo ASS que asigne el valor en la dirección $a_n a la dirección de la variable local creada, donde n es el número de variables locales actual. Aumentar el contador de variables locales.

block

statement



return_stmt



1: Si el scope no es '$global, obtener operando de la pila de operandos y generar cuádruplo que asigna el valor a $r con la operación ASS. Checar si el operando es congruente con el tipo de función actual.

2: Si el scope no es '$global', generar cuádruplo con la operación RTRN. De otra manera, genera el cuádruplo con operación END.

break_stmt

1: Checar si la pila de breaks no está vacía (el estatuto se encuentra dentro de un ciclo). Agregar el contador de cuádruplos actual a la pila de breaks. Generar un cuádruplo GOTO.

print_stmt

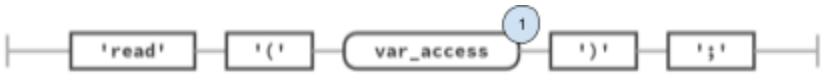
1: Generar cuádruplo PRNT con la dirección de la constante *newline*.

printable

1: Obtener el operando de la pila de operandos. Generar el cuádruplo PRNT.

2: Subir el letrero a la tabla de constantes y obtener dirección. Generar el cuádruplo PRNT.

read_stmt



1: Obtener el operando de la pila de operandos. Checar si el operando es de tipo numérico. Generar cuádruplo READ con la dirección del operando.

load_stmt

1: Subir el letrero a la tabla de constantes y obtener dirección. Generar el cuádruplo LOAD.

save_stmt

1: Subir el letrero a la tabla de constantes y obtener dirección. Generar el cuádruplo SAVE.

resize_stmt

1: Obtener operando de la pila de operandos. Checar si el tipo del operando es numérico. Generar el cuádruplo SETW con la dirección del operando.

2: Obtener operando de la pila de operandos. Checar si el tipo del operando es numérico. Generar el cuádruplo SETH con la dirección del operando.

3: Generar cuádruplo RESZ.

crop_stmt

1: Obtener operando de la pila de operandos. Checar si el tipo del operando es numérico. Generar el cuádruplo SETX con la dirección del operando.

2: Obtener operando de la pila de operandos. Checar si el tipo del operando es numérico. Generar el cuádruplo SETY con la dirección del operando.

3: Obtener operando de la pila de operandos. Checar si el tipo del operando es numérico. Generar el cuádruplo SETW con la dirección del operando.

4: Obtener operando de la pila de operandos. Checar si el tipo del operando es numérico. Generar el cuádruplo SETH con la dirección del operando.

5: Generar cuádruplo CROP.

rotate_stmt

1: Obtener operando de la pila de operandos. Checar si el tipo del operando es numérico. Generar el cuádruplo SETD con la dirección del operando.

2: Generar el cuádruplo ROTA.

flip_v_stmt

1: Generar cuádruplo FLIV.

flip_h_stmt

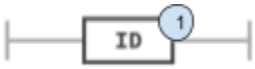
1: 1: Generar cuádruplo FLIH.

var_access

1: Agregar fondo falso a la pila de operadores.

2: Quitar el fondo falso de la pila de operadores.

non_dim_access



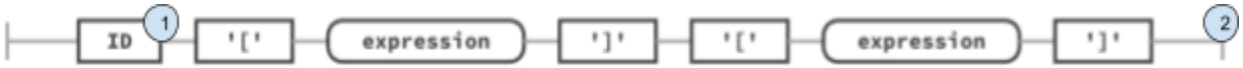
1: Checar si la variable existe en la tabla de variables globales y obtener su información. Si el scope no es '$global', checar la tabla de variables locales primero. También checar si la variable no es dimensionada. Subir la información de tipo y la dirección a la pila de operandos.

arr_access

1: Checar si la variable existe en la tabla de variables globales y obtener su información. Si el scope no es '$global', checar la tabla de variables locales primero. También checar si la variable es un arreglo. Subir la información a la pila de acceso de variables.

2: Obtener operando de la pila de operadores. Checar que sea numérico. Generar el cuádruplo TRUN para truncar el valor. Obtener la infomación de la variable de la pila de acceso de variables. Generar el cuádruplo RANG para verificar el valor. Generar cuádruplos para calcular el número a sumar a la dirección base (base + s_1). Generar el cuádruplo ADDP con un temporal apuntador. Agregar la información de tipo y la dirección del apuntador a la pila de operandos.

mat_access



1: Checar si la variable existe en la tabla de variables globales y obtener su información. Si el scope no es '$global', checar la tabla de variables locales primero. También checar si la variable es una matriz. Subir la información a la pila de acceso de variables.

2: Obtener dos operandos de la pila de operadores. Checar que sean numéricos. Generar los cuádruplos TRUN para truncar los valores. Obtener la infomación de la variable de la pila de acceso de variables. Generar los cuádruplos RANG para verificar los valores. Generar cuádruplos para calcular el número a sumar a la dirección base (base + s_1 * dim_2 + s_2). Generar el cuádruplo ADDP con un temporal apuntador. Agregar la información de tipo y la dirección del apuntador a la pila de operandos.

if_else_stmt

1: Obtener operando de la pila de operandos. Checar si el operando es booleano. Subir el contador de cuádruplos actual a la pila de saltos. Generar cuádruplo GOTF.

2: Obtener el índice del cuádruplo a llenar de la pila de saltos. Subir el contador de cuádruplos actual a la pila de saltos. Generar cuádruplo GOTO. Llenar el cuádruplo con el con el contador de cuádruplos actual.

3: Obtener el índice del cuádruplo a llenar de la pila de saltos. Llenar el cuádruplo con el con el contador de cuádruplos actual.

while_stmt

1: Subir el contador de cuádruplos actual a la pila de saltos. Poner un fondo falso en la pila de breaks.

2: Subir el contador de cuádruplos actual a la pila de saltos. Obtener el operando de la pila de operandos. Checar que el operando sea booleano. Generar el cuádruplo GOTF.

3: Obtener el índice del cuádruplo GOTO a llenar de la pila de saltos. Obtener el índice del cuádruplo de la expresión de la pila de saltos. Generar cuádruplo GOTO con el índice de la expresión. Llenar el cuádruplo GOTO con el contador de cuádruplos actual. Obtener índices de la pila de breaks hasta llegar al fondo falso. Llenar los cuádruplos GOTO en los índices obtenidos con el contador de cuádruplos actual.

for_stmt

1: Poner fondo falso en la pila de breaks.

2: Subir el contador de cuádruplos actual a la pila de saltos.

3: Obtener el operando de la pila de operandos. Checar si es booleano. Subir el contador de cuádruplos actual a la pila de saltos (índice del GOTF).. Generar los cuádruplos GOTF y GOTO.

4: Obtener el índice del cuádruplo GOTF y el índice de la expresión de la pila de saltos. Generar cuádruplo GOTO con el índice de la expresión. Llenar el cuádruplo GOTO el cual se encuentra en el índice del cuádruplo GOTF obtenido + 1. Volver a meter el índice del cuádruplo GOTF a la pila de saltos.

5: Obtener el índice del cuádruplo GOTF de la pila de saltos. Generar cuádruplo GOTO con el índice de la segunda asignación, el cual se encuentra es el índice del cuádruplo GOTF obtenido + 2. Llenar el cuádruplo GOTF con el contador de cuádruplos actual. Obtener índices de la pila de breaks hasta llegar al fondo falso. Llenar los cuádruplos GOTO en los índices obtenidos con el contador de cuádruplos actual.

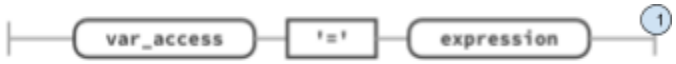
fun_call_stmt

1: Prender la bandera que indica que esta es una llamada de función en un estatuto.

2: Checar la bandera que indica si la última llamada fue una llamada de función void. De no ser así, quitar el operando de la pila de operandos.

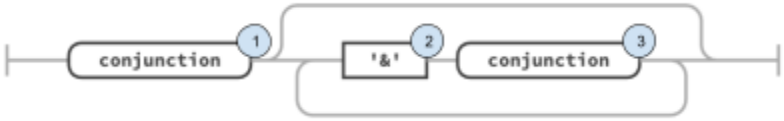
assignment_stmt

assignment



1: Obtener el operando 1 y operando 2 de la pila de operandos. Checar que los operandos sean del mismo tipo. Generar el cuádruplo ASS que asigne el valor del operando 2 al operando 1.

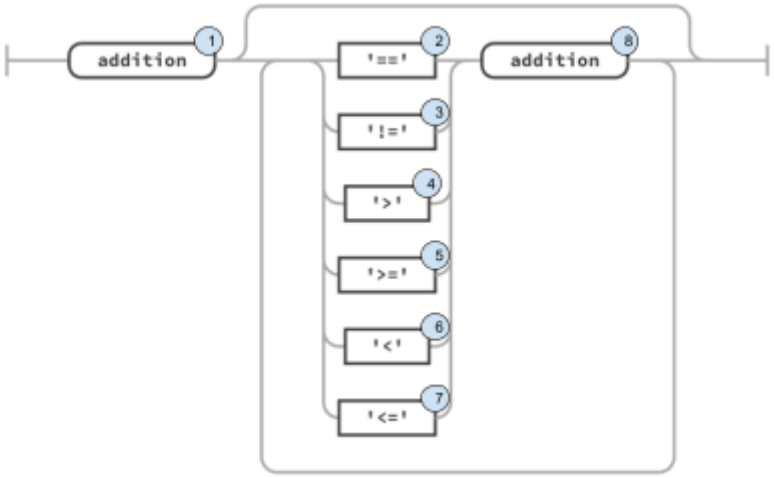
expression



1, 3: Checar si la pila de operadores tiene 'AND' en la cima. Si es así, obtener el operador de la pila de operadores y obtener el operando 1 y el operando 2 de la pila de operandos. Checar que la operación sea válida con el cubo semántico. Obtener una variable temporal. Generar el cuádruplo con la operación obtenida de la pila de operadores, la dirección del operando 1, la dirección del operando 2, y la dirección de la variable temporal. Subir la información de la variable temporal a la pila de operandos.

2: Subir 'AND' a la pila de operadores.

conjunction



1, 8: Checar si la pila de operadores tiene 'EQ', 'NE', 'GT', 'GE', 'LT', o 'LE' en la cima. Si es así, obtener el operador de la pila de operadores y obtener el operando 1 y el operando 2 de la pila de operandos. Checar que la operación sea válida con el cubo semántico. Obtener una variable temporal. Generar el cuádruplo con la operación obtenida de la pila de operadores, la dirección del operando 1, la dirección del operando 2, y la dirección de la variable temporal. Subir la información de la variable temporal a la pila de operandos.

2: Subir 'EQ' a la pila de operadores

3: Subir 'NE' a la pila de operadores

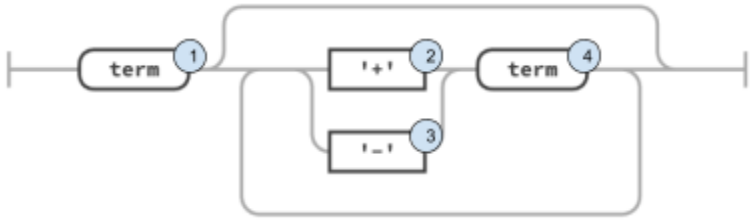
4: Subir 'GT' a la pila de operadores

5: Subir 'GE' a la pila de operadores

6: Subir 'LT' a la pila de operadores

7: Subir 'LE' a la pila de operadores

addition

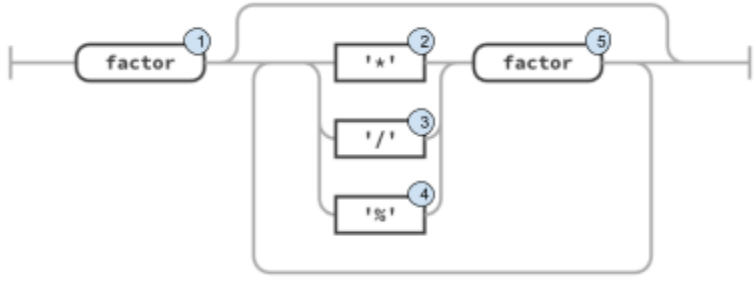


1, 4: Checar si la pila de operadores tiene 'ADD' o 'SUB' en la cima. Si es así, obtener el operador de la pila de operadores y obtener el operando 1 y el operando 2 de la pila de operandos. Checar que la operación sea válida con el cubo semántico. Obtener una variable temporal. Generar el cuádruplo con la operación obtenida de la pila de operadores, la dirección del operando 1, la dirección del operando 2, y la dirección de la variable temporal. Subir la información de la variable temporal a la pila de operandos.

2: Subir 'ADD' a la pila de operadores

3: Subir 'SUB' a la pila de operadores

term



1, 5: Checar si la pila de operadores tiene 'MUL, 'DIV', o 'MOD' en la cima. Si es así, obtener el operador de la pila de operadores y obtener el operando 1 y el operando 2 de la pila de operandos. Checar que la operación sea válida con el cubo semántico. Obtener una variable temporal. Generar el cuádruplo con la operación obtenida de la pila de operadores, la dirección

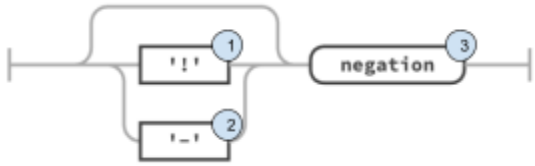
del operando 1, la dirección del operando 2, y la dirección de la variable temporal. Subir la información de la variable temporal a la pila de operandos.

2: Subir 'MUL' a la pila de operadores

3: Subir 'DIV a la pila de operadores

4: Subir 'MOD' a la pila de operadores

factor

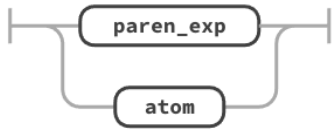


1: Subir 'NOT' a la pila de operadores

2: Subir 'NEG' a la pila de operadores

3: Checar si la pila de operadores tiene 'NOT', o 'NEG' en la cima. Si es así, obtener el operador de la pila de operadores y obtener el operando la pila de operandos. Checar que la operación sea válida con el cubo semántico. Obtener una variable temporal. Generar el cuádruplo con la operación obtenida de la pila de operadores, la dirección del operando, y la dirección de la variable temporal. Subir la información de la variable temporal a la pila de operandos.

negation

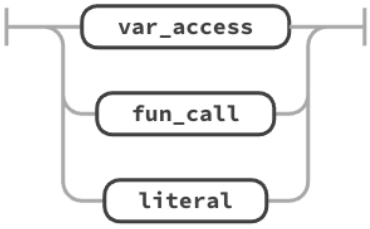


paren_exp

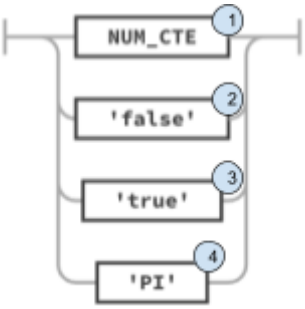
1: Poner un fondo falso en la pila de operadores.

2: Quitar el fondo falso de la pila de operadores.

atom



literal



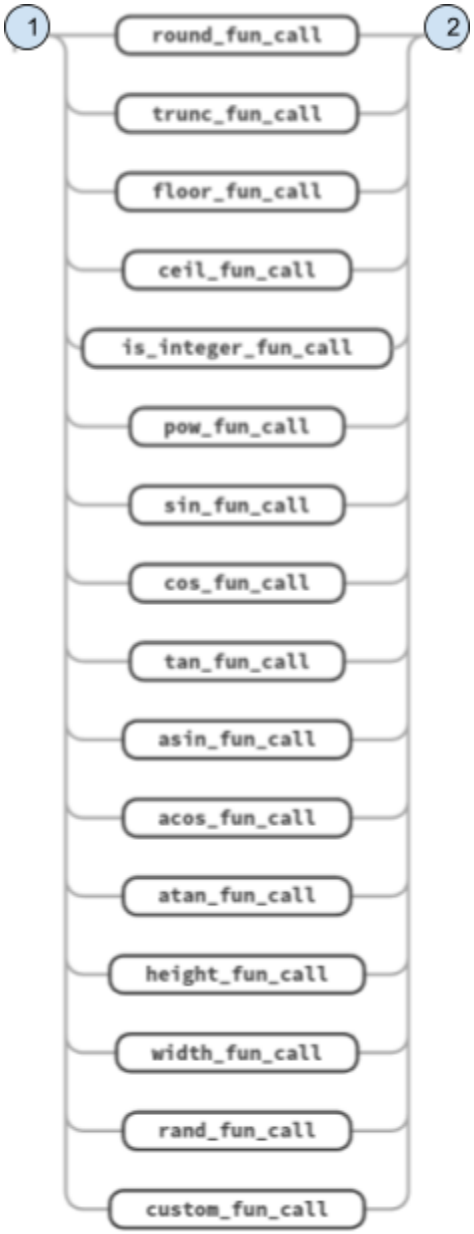
1: Convertir el token a un valor numérico, generar dirección de constante y dar de alta en la tabla de constantes. Subir la dirección a la pila de operandos con un tipo numérico.

2: Subir la dirección $c_f a la pila de operandos con un tipo booleano.

3: Subir la dirección $c_t a la pila de operandos con un tipo booleano.

4: Subir la dirección $c_pi a la pila de operandos con un tipo numérico.

fun_call



1: Agregar fondo falso a la pila de operadores.

2: Quitar fondo falso de la pila de operadores.

round_fun_call

1: Obtener el operando de la pila de operandos. Checar si el operando es numérico. Obtener variable temporal. Generar cuádruplo ROUN con la dirección del operando y la dirección de la

variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

trunc_fun_call

1: Obtener el operando de la pila de operandos. Checar si el operando es numérico. Obtener variable temporal. Generar cuádruplo TRUN con la dirección del operando y la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

ceil_fun_call

1: Obtener el operando de la pila de operandos. Checar si el operando es numérico. Obtener variable temporal. Generar cuádruplo CEIL con la dirección del operando y la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

floor_fun_call

1: Obtener el operando de la pila de operandos. Checar si el operando es numérico. Obtener variable temporal. Generar cuádruplo FLOO con la dirección del operando y la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

is_integer_fun_call

1: Obtener el operando de la pila de operandos. Checar si el operando es numérico. Obtener variable temporal. Generar cuádruplo ISIN con la dirección del operando y la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo booleano.

pow_fun_call

1: Obtener el operando 1 y el operando 2 de la pila de operandos. Checar si los operandos son numéricos. Obtener variable temporal. Generar cuádruplo POW con la dirección del operando 1, la dirección del operando 2 y la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

sin_fun_call

1: Obtener el operando de la pila de operandos. Checar si el operando es numérico. Obtener variable temporal. Generar cuádruplo SIN con la dirección del operando y la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

cos_fun_call

1: Obtener el operando de la pila de operandos. Checar si el operando es numérico. Obtener variable temporal. Generar cuádruplo COS con la dirección del operando y la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

tan_fun_call

1: Obtener el operando de la pila de operandos. Checar si el operando es numérico. Obtener variable temporal. Generar cuádruplo TAN con la dirección del operando y la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

asin_fun_call

1: Obtener el operando de la pila de operandos. Checar si el operando es numérico. Obtener variable temporal. Generar cuádruplo ASIN con la dirección del operando y la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

acos_fun_call

1: Obtener el operando de la pila de operandos. Checar si el operando es numérico. Obtener variable temporal. Generar cuádruplo ACOS con la dirección del operando y la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

atan_fun_call

1: Obtener el operando de la pila de operandos. Checar si el operando es numérico. Obtener variable temporal. Generar cuádruplo ATAN con la dirección del operando y la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

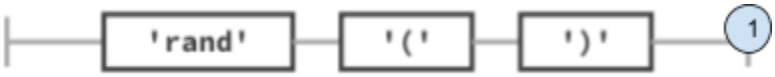
height_fun_call

1: Obtener variable temporal. Generar cuádruplo HEIG con la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

width_fun_call

1: Obtener variable temporal. Generar cuádruplo WIDT con la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

rand_fun_call



1: Obtener variable temporal. Generar cuádruplo RAND con la dirección de la variable temporal. Subir la dirección de la variable temporal a la pila de operandos con un tipo numérico.

custom_fun_call

1: Checar que la función exista. Obtener la información (tipo, parámetros, índice). Checar la bandera que indica que esta es una llamada de función en un estatuto. De ser así resetear la bandera y aceptar que la función sea void. De no ser así rechazar si la función es void. Subir la información de la función junto con un contador de argumentos a la pila de llamadas de funciones.

2, 3: Obtener la información de la función de la cima de la pila de llamadas de funciones (con un peek). Checar que el contador de argumentos no sea mayor a la cantidad de parámetros. Obtener el operando de la pila de operandos y checar que su tipo coincida con el parámetro correspondiente. Generar el cuádruplo ASS que asigne el valor en la dirección del operando a la dirección $a_n donde n es el contador de argumentos actual. Aumentar el valor del contador de argumentos en la cima de la pila de llamadas de funciones.

4: Obtener la información de la función de la pila de llamadas de funciones (con un pop). Checar que no haya parámetros faltantes. Generar un cuádruplo CALL con el índice de la función. Si la función fue una función void, prender la bandera que indica que la última llama de función fue una función void y terminar. De no ser así, apagar la bandera, y obtener una variable temporal. Generar un cuádruplo ASS que asigne el valor en $r a la dirección de la variable temporal, y subir la dirección a la pila de operandos junto con el tipo de la función.

Tabla semántica

Operaciones de dos operandos

op 1:	number	number	boolean	boolean
op 2:	number	boolean	number	boolean
ADD	number	N/A	N/A	N/A
SUB	number	N/A	N/A	N/A
MUL	number	N/A	N/A	N/A
DIV	number	N/A	N/A	N/A
MOD	number	N/A	N/A	N/A
EQ	boolean	N/A	N/A	boolean

NE	boolean	N/A	N/A	boolean
GT	boolean	N/A	N/A	N/A
GE	boolean	N/A	N/A	N/A
LT	boolean	N/A	N/A	N/A
LE	boolean	N/A	N/A	N/A
OR	N/A	N/A	N/A	boolean
AND	N/A	N/A	N/A	boolean

Operaciones de un operando (unarias)

op:	number	boolean
NOT	N/A	boolean
NEG	number	N/A

Estructuras usadas

Para el almacenamiento de los cuádruplos, se utilizó un arreglo dinámico (vector), ya que se necesitaba una estructura que fuera creciendo durante la ejecución, y que permitiera acceso a un elemento que se encontrara en un índice arbitrario (para la generación de estatutos no lineales como GOTO).

Para la tabla de funciones se utilizó un objeto de JavaScript, el cual funciona como un diccionario. Se utilizó para poder indexar un valor nuevo durante la ejecución, usando el nombre de la función como llave. Este diccionario es sí almacena un objeto que almacena el tipo de la función, su lista de parámetros (vector), y el índice del cuádruplo donde inicia.

Similarmente, se hizo lo mismo para la tabla de variables locales y globales, pero guardando un objeto que almacena su tipo, la lista de dimensiones (vector de números), y su dirección. Sólo se ocupó una sola tabla de variables locales la cuál es reutilizada para cada función.

Se utilizaron contadores (variables numéricas) para llevar un conteo de las variables globales, locales, temporales globales, temporales globales, apuntadores globales, apuntadores locales, y constantes.

Se decidió utilizar una pila para mantener un control de los operadores, operandos, saltos, breaks, llamadas de funciones, y acceso a variables. Se eligió una pila porque se requería una estructura que fuera creciendo y que permitiera acceso fácil al último elemento agregado.

Para las constantes se usaron tres diccionarios. Uno que usó como llave la dirección de la constante y contiene su valor (puede ser numérico, booleano, o string). Este es el diccionario

que es exportado en el código objeto. El segundo diccionario se usa durante la compilación, usa como llave un valor numérico (convertido a string) y contiene la dirección asociada con ese número. El tercer diccionario es similar al segundo, pero es usado para las constantes de tipo string. Se eligió usar un diccionario porque facilita la indexación y la obtención de datos fácil y rápidamente utilizando strings.

Codigo objeto

El código objeto generado es simplemente un archivo JSON que contiene una representación de un objeto JS con dos propiedades:

- 'quadruples', la cual contiene el arreglo de cuádruplos (los cuales son arreglos de longitud 4.
- 'constTable', la cual contiene un objeto (diccionario) con direcciones virtuales como llaves y contiene los valores constantes los cuales pueden ser booleanos, numéricos, o strings.

Por lo tanto, un ejemplo de un archivo objeto quedaría así:

```
{
    "quadruples": [
        [ "GOTO",  null,   null,  "$c_1" ],
        [ "ADD",  "$c_1", "$c_2", "$t_0" ],
        [ "ASS",  "$t_0",  null,  "$g_0" ],
        [ "PRNT", "$g_0",  null,   null  ],
        [ "PRNT", "$c_n",  null,   null  ],
        [ "END",   null,   null,   null  ]
    ],
    "constTable": {
        "$c_f": false,
        "$c_t": true,
        "$c_n": "\n",
        "$c_0": 0,
        "$c_pi": 3.141592653589793,
        "$c_1": 1,
        "$c_2": 2
    }
}
```

Descripción de la máquina virtual

La máquina virtual fue hecha con las mismas herramientas que el compilador. Se utilizó la librería Sharp (https://www.npmjs.com/package/sharp) para poder hacer las operaciones de imágenes.

Para la memoria durante la ejecución, se utilizó un objeto de JavaScript, el cual funciona como un diccionario. Se decidió utilizar esto ya que las direcciones virtuales están descritas por

strings, y esta estructura permite fácilmente indexar y recuperar la información almacenada. Se sabe que esto causa que se use una mayor cantidad de memoria, y es un sacrificio que se hizo para simplificar la codificación.

La máquina virtual usa un diccionario para la memoria global, la cual contiene las variables globales, las constantes, los temporales globales, las variables para pasar argumentos a funciones, y la dirección que usan las funciones para regresar un valor. Para la memoria virtual, se tiene una pila que contiene memorias locales (también diccionarios), la cual empieza vacía y a la cual se le agrega un elemento nuevo cuando se ejecuta la instrucción CALL. Cuando se requiere recuperar un valor de la memoria virtual, se hace un peek() a esta pila y se obtiene el valor del objeto obtenido. Cuando se ejecuta la instrucción RTRN, se le hace un pop() a esta pila.

También se utiliza una pila para guardar el pointer de ejecución cada vez que se crea una nueva memoria con la instrucción CALL, y este pointer es recuperado y restablecido cuando se ejecuta la instrucción RTRN.

Pruebas de funcionamiento

Prueba de estatutos no lineales

Entrada

```
program forprog:

var number i, j;

main:
{
    for (i = 0; i < 10; i = i + 1) {
        if (i == 7) {
            print("found seven");
            break;
            break;
            break;
        }

        if (false) {
            break;
        }

        j = 0;
        while (j < 10) {
            if (j == 7) {
                print("found seven");
                break;
                break;
```

```
            }
            if (false | false) {
                break;
            }
            print("i, j: ", i, ", ", j);
            j = j + 1;
        }
    }
}
```

Código intermedio

```
{
    "quadruples": [
        [
            "GOTO",
            null,
            null,
            "$c_1"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$g_0"
        ],
        [
            "LT",
            "$g_0",
            "$c_2",
            "$t_0"
        ],
        [
            "GOTF",
            "$t_0",
            null,
            "$c_16"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_4"
        ],
        [
            "ADD",
            "$g_0",
            "$c_1",
            "$t_0"
        ],
```

```
        [
            "ASS",
            "$t_0",
            null,
            "$g_0"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_3"
        ],
        [
            "EQ",
            "$g_0",
            "$c_5",
            "$t_0"
        ],
        [
            "GOTF",
            "$t_0",
            null,
            "$c_7"
        ],
        [
            "PRNT",
            "$c_6",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "GOTO",
            null,
            null,
            "$c_16"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_16"
        ],
        [
            "GOTO",
            null,
```

```
            null,
            "$c_16"
        ],
        [
            "GOTF",
            "$c_f",
            null,
            "$c_8"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_16"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$g_1"
        ],
        [
            "LT",
            "$g_1",
            "$c_2",
            "$t_0"
        ],
        [
            "GOTF",
            "$t_0",
            null,
            "$c_14"
        ],
        [
            "EQ",
            "$g_1",
            "$c_5",
            "$t_0"
        ],
        [
            "GOTF",
            "$t_0",
            null,
            "$c_9"
        ],
        [
            "PRNT",
            "$c_6",
            null,
            null
        ],
```

```
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "GOTO",
            null,
            null,
            "$c_14"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_14"
        ],
        [
            "OR",
            "$c_f",
            "$c_f",
            "$t_0"
        ],
        [
            "GOTF",
            "$t_0",
            null,
            "$c_10"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_14"
        ],
        [
            "PRNT",
            "$c_11",
            null,
            null
        ],
        [
            "PRNT",
            "$g_0",
            null,
            null
        ],
        [
            "PRNT",
            "$c_12",
```

```
            null,
            null
        ],
        [
            "PRNT",
            "$g_1",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "ADD",
            "$g_1",
            "$c_1",
            "$t_0"
        ],
        [
            "ASS",
            "$t_0",
            null,
            "$g_1"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_13"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_15"
        ],
        [
            "END",
            null,
            null,
            "$c_0"
        ]
    ],
    "constTable": {
        "$c_f": false,
        "$c_t": true,
        "$c_n": "\n",
        "$c_0": 0,
```

```
        "$c_pi": 3.141592653589793,
        "$c_1": 1,
        "$c_2": 10,
        "$c_3": 2,
        "$c_4": 8,
        "$c_5": 7,
        "$c_6": "found seven",
        "$c_7": 15,
        "$c_8": 17,
        "$c_9": 26,
        "$c_10": 29,
        "$c_11": "i, j: ",
        "$c_12": ", ",
        "$c_13": 18,
        "$c_14": 37,
        "$c_15": 5,
        "$c_16": 38
    }
}
```

Salida

```
i, j: 0, 0
i, j: 0, 1
i, j: 0, 2
i, j: 0, 3
i, j: 0, 4
i, j: 0, 5
i, j: 0, 6
found seven
i, j: 1, 0
i, j: 1, 1
i, j: 1, 2
i, j: 1, 3
i, j: 1, 4
i, j: 1, 5
i, j: 1, 6
found seven
i, j: 2, 0
i, j: 2, 1
i, j: 2, 2
i, j: 2, 3
i, j: 2, 4
i, j: 2, 5
i, j: 2, 6
found seven
i, j: 3, 0
i, j: 3, 1
i, j: 3, 2
i, j: 3, 3
```

```
i, j: 3, 4
i, j: 3, 5
i, j: 3, 6
found seven
i, j: 4, 0
i, j: 4, 1
i, j: 4, 2
i, j: 4, 3
i, j: 4, 4
i, j: 4, 5
i, j: 4, 6
found seven
i, j: 5, 0
i, j: 5, 1
i, j: 5, 2
i, j: 5, 3
i, j: 5, 4
i, j: 5, 5
i, j: 5, 6
found seven
i, j: 6, 0
i, j: 6, 1
i, j: 6, 2
i, j: 6, 3
i, j: 6, 4
i, j: 6, 5
i, j: 6, 6
found seven
found seven
```

Prueba de arreglos y matrices

Entrada

```
program mats:

var number[10][7] mat;
var number a, b;

function void fillMat(number start, number step):
var number i,j;
{
    for (i = 0; i < 10; i = i + 1) {
        for (j = 0; j < 7; j = j + 1) {
            mat[i][j] = start;
            start = start + step;
        }
    }
}
```

```
function void printMat():
var number i,j;
{
    for (i = 0; i < 10; i = i + 1) {
        for (j = 0; j < 7; j = j + 1) {
            print(mat[i][j]);
        }
        print("");
    }
}

function number zero(number z):
{
    return 0;
}

main:

var number[1] p;
{
    p[0] = 0;
    fillMat(0, 1);
    printMat();
    mat[5][5] = -1;
    mat [ p[0] ] [ p[ p[zero(zero(zero(p[p[0]])))] ] ] = -10;
    print("--------------------");
    printMat();
}
```

Código intermedio

```
{
    "quadruples": [
        [
            "GOTO",
            null,
            null,
            "$c_19"
        ],
        [
            "ASS",
            "$a_0",
            null,
            "$l_0"
        ],
        [
            "ASS",
            "$a_1",
            null,
```

```
            "$l_1"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$l_2"
        ],
        [
            "LT",
            "$l_2",
            "$c_1",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_9"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_1"
        ],
        [
            "ADD",
            "$l_2",
            "$c_3",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$l_2"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_4"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$l_3"
        ],
        [
```

```
            "LT",
            "$l_3",
            "$c_2",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_8"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_6"
        ],
        [
            "ADD",
            "$l_3",
            "$c_3",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$l_3"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_5"
        ],
        [
            "TRUN",
            "$l_2",
            null,
            "$lt_0"
        ],
        [
            "RANG",
            "$lt_0",
            "$c_0",
            "$c_1"
        ],
        [
            "TRUN",
            "$l_3",
            null,
```

```
            "$lt_1"
        ],
        [
            "RANG",
            "$lt_1",
            "$c_0",
            "$c_2"
        ],
        [
            "MUL",
            "$lt_0",
            "$c_2",
            "$lt_2"
        ],
        [
            "ADD",
            "$lt_2",
            "$lt_1",
            "$lt_0"
        ],
        [
            "ADDP",
            "$g_0",
            "$lt_0",
            "$l$_0"
        ],
        [
            "ASS",
            "$l_0",
            null,
            "$l$_0"
        ],
        [
            "ADD",
            "$l_0",
            "$l_1",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$l_0"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_7"
        ],
        [
```

```
            "GOTO",
            null,
            null,
            "$c_2"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$l_0"
        ],
        [
            "LT",
            "$l_0",
            "$c_1",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_18"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_11"
        ],
        [
            "ADD",
            "$l_0",
            "$c_3",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$l_0"
        ],
        [
            "GOTO",
            null,
            null,
```

```
            "$c_10"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$l_1"
        ],
        [
            "LT",
            "$l_1",
            "$c_2",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_15"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_13"
        ],
        [
            "ADD",
            "$l_1",
            "$c_3",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$l_1"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_12"
        ],
        [
            "TRUN",
            "$l_0",
            null,
            "$lt_0"
        ],
        [
```

```
            "RANG",
            "$lt_0",
            "$c_0",
            "$c_1"
        ],
        [
            "TRUN",
            "$l_1",
            null,
            "$lt_1"
        ],
        [
            "RANG",
            "$lt_1",
            "$c_0",
            "$c_2"
        ],
        [
            "MUL",
            "$lt_0",
            "$c_2",
            "$lt_2"
        ],
        [
            "ADD",
            "$lt_2",
            "$lt_1",
            "$lt_0"
        ],
        [
            "ADDP",
            "$g_0",
            "$lt_0",
            "$l$_0"
        ],
        [
            "PRNT",
            "$l$_0",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "GOTO",
            null,
            null,
```

```
            "$c_14"
        ],
        [
            "PRNT",
            "$c_16",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "GOTO",
            null,
            null,
            "$c_17"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
            "$a_0",
            null,
            "$l_0"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$r"
        ],
        [
```

```
            "RTRN",
            null,
            null,
            null
        ],
        [
            "TRUN",
            "$c_0",
            null,
            "$t_0"
        ],
        [
            "RANG",
            "$t_0",
            "$c_0",
            "$c_3"
        ],
        [
            "ADDP",
            "$g_72",
            "$t_0",
            "$$_0"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$$_0"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$a_0"
        ],
        [
            "ASS",
            "$c_3",
            null,
            "$a_1"
        ],
        [
            "CALL",
            null,
            null,
            "$c_3"
        ],
        [
            "CALL",
            null,
            null,
```

```
            "$c_20"
        ],
        [
            "TRUN",
            "$c_21",
            null,
            "$t_0"
        ],
        [
            "RANG",
            "$t_0",
            "$c_0",
            "$c_1"
        ],
        [
            "TRUN",
            "$c_21",
            null,
            "$t_1"
        ],
        [
            "RANG",
            "$t_1",
            "$c_0",
            "$c_2"
        ],
        [
            "MUL",
            "$t_0",
            "$c_2",
            "$t_2"
        ],
        [
            "ADD",
            "$t_2",
            "$t_1",
            "$t_0"
        ],
        [
            "ADDP",
            "$g_0",
            "$t_0",
            "$$_0"
        ],
        [
            "NEG",
            "$c_3",
            null,
            "$t_0"
        ],
        [
```

```
            "ASS",
            "$t_0",
            null,
            "$$_0"
        ],
        [
            "TRUN",
            "$c_0",
            null,
            "$t_1"
        ],
        [
            "RANG",
            "$t_1",
            "$c_0",
            "$c_3"
        ],
        [
            "ADDP",
            "$g_72",
            "$t_1",
            "$$_0"
        ],
        [
            "TRUN",
            "$c_0",
            null,
            "$t_2"
        ],
        [
            "RANG",
            "$t_2",
            "$c_0",
            "$c_3"
        ],
        [
            "ADDP",
            "$g_72",
            "$t_2",
            "$$_1"
        ],
        [
            "TRUN",
            "$$_1",
            null,
            "$t_0"
        ],
        [
            "RANG",
            "$t_0",
            "$c_0",
```

```
            "$c_3"
        ],
        [
            "ADDP",
            "$g_72",
            "$t_0",
            "$$_2"
        ],
        [
            "ASS",
            "$$_2",
            null,
            "$a_0"
        ],
        [
            "CALL",
            null,
            null,
            "$c_22"
        ],
        [
            "ASS",
            "$r",
            null,
            "$t_1"
        ],
        [
            "ASS",
            "$t_1",
            null,
            "$a_0"
        ],
        [
            "CALL",
            null,
            null,
            "$c_22"
        ],
        [
            "ASS",
            "$r",
            null,
            "$t_2"
        ],
        [
            "ASS",
            "$t_2",
            null,
            "$a_0"
        ],
        [
```

```
            "CALL",
            null,
            null,
            "$c_22"
        ],
        [
            "ASS",
            "$r",
            null,
            "$t_0"
        ],
        [
            "TRUN",
            "$t_0",
            null,
            "$t_1"
        ],
        [
            "RANG",
            "$t_1",
            "$c_0",
            "$c_3"
        ],
        [
            "ADDP",
            "$g_72",
            "$t_1",
            "$$_1"
        ],
        [
            "TRUN",
            "$$_1",
            null,
            "$t_2"
        ],
        [
            "RANG",
            "$t_2",
            "$c_0",
            "$c_3"
        ],
        [
            "ADDP",
            "$g_72",
            "$t_2",
            "$$_2"
        ],
        [
            "TRUN",
            "$$_0",
            null,
```

```
            "$t_1"
        ],
        [
            "RANG",
            "$t_1",
            "$c_0",
            "$c_1"
        ],
        [
            "TRUN",
            "$$_2",
            null,
            "$t_0"
        ],
        [
            "RANG",
            "$t_0",
            "$c_0",
            "$c_2"
        ],
        [
            "MUL",
            "$t_1",
            "$c_2",
            "$t_2"
        ],
        [
            "ADD",
            "$t_2",
            "$t_0",
            "$t_1"
        ],
        [
            "ADDP",
            "$g_0",
            "$t_1",
            "$$_1"
        ],
        [
            "NEG",
            "$c_1",
            null,
            "$t_1"
        ],
        [
            "ASS",
            "$t_1",
            null,
            "$$_1"
        ],
        [
```

```
            "PRNT",
            "$c_23",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "CALL",
            null,
            null,
            "$c_20"
        ],
        [
            "END",
            null,
            null,
            "$c_0"
        ]
    ],
    "constTable": {
        "$c_f": false,
        "$c_t": true,
        "$c_n": "\n",
        "$c_0": 0,
        "$c_pi": 3.141592653589793,
        "$c_1": 10,
        "$c_2": 7,
        "$c_3": 1,
        "$c_4": 4,
        "$c_5": 11,
        "$c_6": 17,
        "$c_7": 14,
        "$c_8": 28,
        "$c_9": 29,
        "$c_10": 31,
        "$c_11": 37,
        "$c_12": 38,
        "$c_13": 44,
        "$c_14": 41,
        "$c_15": 54,
        "$c_16": "",
        "$c_17": 34,
        "$c_18": 57,
        "$c_19": 63,
        "$c_20": 30,
        "$c_21": 5,
```

```
        "$c_22": 58,
        "$c_23": "--------------------"
    }
}
```

Salida

```
0
1
2
3
4
5
6

7
8
9
10
11
12
13

14
15
16
17
18
19
20

21
22
23
24
25
26
27

28
29
30
31
32
33
34

35
36
37
```

38
39
40
41

42
43
44
45
46
47
48

49
50
51
52
53
54
55

56
57
58
59
60
61
62

63
64
65
66
67
68
69

-10
1
2
3
4
5
6

7
8
9
10
11

12
13

14
15
16
17
18
19
20

21
22
23
24
25
26
27

28
29
30
31
32
33
34

35
36
37
38
39
-1
41

42
43
44
45
46
47
48

49
50
51
52
53
54
55

```
56
57
58
59
60
61
62

63
64
65
66
67
68
69
```

Prueba de funciones y recursividad

Entrada

```
program funcalls:

var number z;

function number fibonacci(number n):
{
    if (n <= 0) {
        return 0;
    }

    if (n <= 1) {
        return 1;
    }

    return fibonacci(n - 2) + fibonacci(n - 1);
}

function number factorial_recursive(number n):
{
    if (n <= 0) {
        return 1;
    }
    return n * factorial_recursive(n - 1);
}

function void print_sequence(number n):
var number i;
{
```

```
    for (i = 0; i <= n; i = i + 1) {
        print(i);
    }
}

function number add(number a, number b, number c, number d):
{
    if (d != 0) {
        return add(a, b, c + d, 0);
    }
    if (c != 0) {
        return add(a, b + c, 0, 0);
    }
    if (b != 0) {
        return add(a + b, 0, 0, 0);
    }
    return a;
}

function void print_sequence_sequence(number n):
var number i;
{
    for (i = 0; i <= n; i = i + 1) {
        print_sequence(i);
        print("");
    }
}

function number factorial_iterative(number n):
var number output;
{
    output = 1;
    while (n > 0) {
        output = output * n;
        n = n - 1;
    }
    return output;
}

main:
var number a, b, c, d, n;
{
    n = 5;
    #print("enter n:");
    #read(n);

    print_sequence_sequence(n);

    print(n, " factorial (iterative): ", factorial_iterative(n));
    print(n, " factorial (recursive): ", factorial_recursive(n));
```

```
    a = rand() * 10;
    b = rand() * 10;
    c = rand() * 10;
    d = rand() * 10;
    print(a, " + ", b, " + ", c, " + ", d, " = ", add(a,b,c,d));
}
```

Código intermedio


```
{
    "quadruples": [
        [
            "GOTO",
            null,
            null,
            "$c_23"
        ],
        [
            "ASS",
            "$a_0",
            null,
            "$l_0"
        ],
        [
            "LE",
            "$l_0",
            "$c_0",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_1"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "LE",
```

```
            "$l_0",
            "$c_2",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_3"
        ],
        [
            "ASS",
            "$c_2",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "SUB",
            "$l_0",
            "$c_4",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$a_0"
        ],
        [
            "CALL",
            null,
            null,
            "$c_2"
        ],
        [
            "ASS",
            "$r",
            null,
            "$lt_0"
        ],
        [
            "SUB",
            "$l_0",
            "$c_2",
            "$lt_1"
```

```
        ],
        [
            "ASS",
            "$lt_1",
            null,
            "$a_0"
        ],
        [
            "CALL",
            null,
            null,
            "$c_2"
        ],
        [
            "ASS",
            "$r",
            null,
            "$lt_1"
        ],
        [
            "ADD",
            "$lt_0",
            "$lt_1",
            "$lt_2"
        ],
        [
            "ASS",
            "$lt_2",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
```

```
            "$a_0",
            null,
            "$l_0"
        ],
        [
            "LE",
            "$l_0",
            "$c_0",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_5"
        ],
        [
            "ASS",
            "$c_2",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "SUB",
            "$l_0",
            "$c_2",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$a_0"
        ],
        [
            "CALL",
            null,
            null,
            "$c_6"
        ],
        [
            "ASS",
            "$r",
            null,
            "$lt_0"
```

```
        ],
        [
            "MUL",
            "$l_0",
            "$lt_0",
            "$lt_1"
        ],
        [
            "ASS",
            "$lt_1",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
            "$a_0",
            null,
            "$l_0"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$l_1"
        ],
        [
            "LE",
            "$l_1",
            "$l_0",
            "$lt_0"
        ],
        [
            "GOTF",
```

```
            "$lt_0",
            null,
            "$c_10"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_8"
        ],
        [
            "ADD",
            "$l_1",
            "$c_2",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$l_1"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_7"
        ],
        [
            "PRNT",
            "$l_1",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "GOTO",
            null,
            null,
            "$c_9"
        ],
        [
            "RTRN",
            null,
            null,
            null
```

```
        ],
        [
            "ASS",
            "$a_0",
            null,
            "$l_0"
        ],
        [
            "ASS",
            "$a_1",
            null,
            "$l_1"
        ],
        [
            "ASS",
            "$a_2",
            null,
            "$l_2"
        ],
        [
            "ASS",
            "$a_3",
            null,
            "$l_3"
        ],
        [
            "NE",
            "$l_3",
            "$c_0",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_12"
        ],
        [
            "ASS",
            "$l_0",
            null,
            "$a_0"
        ],
        [
            "ASS",
            "$l_1",
            null,
            "$a_1"
        ],
        [
            "ADD",
```

```
            "$l_2",
            "$l_3",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$a_2"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$a_3"
        ],
        [
            "CALL",
            null,
            null,
            "$c_11"
        ],
        [
            "ASS",
            "$r",
            null,
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "NE",
            "$l_2",
            "$c_0",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_13"
```

```
        ],
        [
            "ASS",
            "$l_0",
            null,
            "$a_0"
        ],
        [
            "ADD",
            "$l_1",
            "$l_2",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$a_1"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$a_2"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$a_3"
        ],
        [
            "CALL",
            null,
            null,
            "$c_11"
        ],
        [
            "ASS",
            "$r",
            null,
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$r"
        ],
        [
            "RTRN",
```

```
            null,
            null,
            null
        ],
        [
            "NE",
            "$l_1",
            "$c_0",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_14"
        ],
        [
            "ADD",
            "$l_0",
            "$l_1",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$a_0"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$a_1"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$a_2"
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$a_3"
        ],
        [
            "CALL",
            null,
            null,
            "$c_11"
```

```
        ],
        [
            "ASS",
            "$r",
            null,
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
            "$l_0",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
            "$a_0",
            null,
            "$l_0"
        ],
        [
            "ASS",
```

```
            "$c_0",
            null,
            "$l_1"
        ],
        [
            "LE",
            "$l_1",
            "$l_0",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_20"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_16"
        ],
        [
            "ADD",
            "$l_1",
            "$c_2",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$l_1"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_15"
        ],
        [
            "ASS",
            "$l_1",
            null,
            "$a_0"
        ],
        [
            "CALL",
            null,
            null,
            "$c_17"
```

```
        ],
        [
            "PRNT",
            "$c_18",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "GOTO",
            null,
            null,
            "$c_19"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
            "$a_0",
            null,
            "$l_0"
        ],
        [
            "ASS",
            "$c_2",
            null,
            "$l_1"
        ],
        [
            "GT",
            "$l_0",
            "$c_0",
            "$lt_0"
        ],
        [
            "GOTF",
            "$lt_0",
            null,
            "$c_22"
        ],
        [
            "MUL",
```

```
            "$l_1",
            "$l_0",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$l_1"
        ],
        [
            "SUB",
            "$l_0",
            "$c_2",
            "$lt_0"
        ],
        [
            "ASS",
            "$lt_0",
            null,
            "$l_0"
        ],
        [
            "GOTO",
            null,
            null,
            "$c_21"
        ],
        [
            "ASS",
            "$l_1",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
        ],
        [
            "ASS",
            "$c_0",
            null,
            "$r"
        ],
        [
            "RTRN",
            null,
            null,
            null
```

```
        ],
        [
            "ASS",
            "$c_24",
            null,
            "$g_5"
        ],
        [
            "ASS",
            "$g_5",
            null,
            "$a_0"
        ],
        [
            "CALL",
            null,
            null,
            "$c_25"
        ],
        [
            "PRNT",
            "$g_5",
            null,
            null
        ],
        [
            "PRNT",
            "$c_26",
            null,
            null
        ],
        [
            "ASS",
            "$g_5",
            null,
            "$a_0"
        ],
        [
            "CALL",
            null,
            null,
            "$c_27"
        ],
        [
            "ASS",
            "$r",
            null,
            "$t_0"
        ],
        [
            "PRNT",
```

```
            "$t_0",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "PRNT",
            "$g_5",
            null,
            null
        ],
        [
            "PRNT",
            "$c_28",
            null,
            null
        ],
        [
            "ASS",
            "$g_5",
            null,
            "$a_0"
        ],
        [
            "CALL",
            null,
            null,
            "$c_6"
        ],
        [
            "ASS",
            "$r",
            null,
            "$t_0"
        ],
        [
            "PRNT",
            "$t_0",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
```

```
        ],
        [
            "RAND",
            null,
            null,
            "$t_0"
        ],
        [
            "MUL",
            "$t_0",
            "$c_3",
            "$t_1"
        ],
        [
            "ASS",
            "$t_1",
            null,
            "$g_1"
        ],
        [
            "RAND",
            null,
            null,
            "$t_0"
        ],
        [
            "MUL",
            "$t_0",
            "$c_3",
            "$t_1"
        ],
        [
            "ASS",
            "$t_1",
            null,
            "$g_2"
        ],
        [
            "RAND",
            null,
            null,
            "$t_0"
        ],
        [
            "MUL",
            "$t_0",
            "$c_3",
            "$t_1"
        ],
        [
            "ASS",
```

```
            "$t_1",
            null,
            "$g_3"
        ],
        [
            "RAND",
            null,
            null,
            "$t_0"
        ],
        [
            "MUL",
            "$t_0",
            "$c_3",
            "$t_1"
        ],
        [
            "ASS",
            "$t_1",
            null,
            "$g_4"
        ],
        [
            "PRNT",
            "$g_1",
            null,
            null
        ],
        [
            "PRNT",
            "$c_29",
            null,
            null
        ],
        [
            "PRNT",
            "$g_2",
            null,
            null
        ],
        [
            "PRNT",
            "$c_29",
            null,
            null
        ],
        [
            "PRNT",
            "$g_3",
            null,
            null
```

```
        ],
        [
            "PRNT",
            "$c_29",
            null,
            null
        ],
        [
            "PRNT",
            "$g_4",
            null,
            null
        ],
        [
            "PRNT",
            "$c_30",
            null,
            null
        ],
        [
            "ASS",
            "$g_1",
            null,
            "$a_0"
        ],
        [
            "ASS",
            "$g_2",
            null,
            "$a_1"
        ],
        [
            "ASS",
            "$g_3",
            null,
            "$a_2"
        ],
        [
            "ASS",
            "$g_4",
            null,
            "$a_3"
        ],
        [
            "CALL",
            null,
            null,
            "$c_11"
        ],
        [
            "ASS",
```

```
            "$r",
            null,
            "$t_0"
        ],
        [
            "PRNT",
            "$t_0",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "END",
            null,
            null,
            "$c_0"
        ]
    ],
    "constTable": {
        "$c_f": false,
        "$c_t": true,
        "$c_n": "\n",
        "$c_0": 0,
        "$c_pi": 3.141592653589793,
        "$c_1": 6,
        "$c_2": 1,
        "$c_3": 10,
        "$c_4": 2,
        "$c_5": 28,
        "$c_6": 23,
        "$c_7": 39,
        "$c_8": 45,
        "$c_9": 42,
        "$c_10": 48,
        "$c_11": 49,
        "$c_12": 64,
        "$c_13": 75,
        "$c_14": 86,
        "$c_15": 92,
        "$c_16": 98,
        "$c_17": 37,
        "$c_18": "",
        "$c_19": 95,
        "$c_20": 103,
        "$c_21": 106,
        "$c_22": 113,
```

```
        "$c_23": 117,
        "$c_24": 5,
        "$c_25": 90,
        "$c_26": " factorial (iterative): ",
        "$c_27": 104,
        "$c_28": " factorial (recursive): ",
        "$c_29": " + ",
        "$c_30": " = "
    }
}
```

Salida

```
0

0
1

0
1
2

0
1
2
3

0
1
2
3
4

0
1
2
3
4
5

5 factorial (iterative): 120
5 factorial (recursive): 120
4.027021142448111 + 8.617829999129487 + 9.917784065180417 +
4.8895327576046 = 27.452167964362612
```

Prueba de manipulación de imágen

Entrada

```
program image:

main:
{
    load("test.png");

    print("height: ", getHeight());
    print("width: ", getWidth());

    #crop(100, 0, 100, 100);
    rotate(90);
    #flipVertically();
    flipHorizontally();

    print("new height: ", getHeight());
    print("new width: ", getWidth());

    save("output.png");
}
```

test.png

Código intermedio

```
{
    "quadruples": [
        [
            "GOTO",
            null,
```

```
            null,
            "$c_1"
        ],
        [
            "LOAD",
            "$c_2",
            null,
            null
        ],
        [
            "PRNT",
            "$c_3",
            null,
            null
        ],
        [
            "HEIG",
            null,
            null,
            "$t_0"
        ],
        [
            "PRNT",
            "$t_0",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "PRNT",
            "$c_4",
            null,
            null
        ],
        [
            "WIDT",
            null,
            null,
            "$t_0"
        ],
        [
            "PRNT",
            "$t_0",
            null,
            null
        ],
```

```
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "SETD",
            "$c_5",
            null,
            null
        ],
        [
            "ROTA",
            null,
            null,
            null
        ],
        [
            "FLIH",
            null,
            null,
            null
        ],
        [
            "PRNT",
            "$c_6",
            null,
            null
        ],
        [
            "HEIG",
            null,
            null,
            "$t_0"
        ],
        [
            "PRNT",
            "$t_0",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "PRNT",
            "$c_7",
```

```
            null,
            null
        ],
        [
            "WIDT",
            null,
            null,
            "$t_0"
        ],
        [
            "PRNT",
            "$t_0",
            null,
            null
        ],
        [
            "PRNT",
            "$c_n",
            null,
            null
        ],
        [
            "SAVE",
            "$c_8",
            null,
            null
        ],
        [
            "END",
            null,
            null,
            "$c_0"
        ]
    ],
    "constTable": {
        "$c_f": false,
        "$c_t": true,
        "$c_n": "\n",
        "$c_0": 0,
        "$c_pi": 3.141592653589793,
        "$c_1": 1,
        "$c_2": "test.png",
        "$c_3": "height: ",
        "$c_4": "width: ",
        "$c_5": 90,
        "$c_6": "new height: ",
        "$c_7": "new width: ",
        "$c_8": "output.png"
    }
}
```

Salida

output.png

Manual de usuario

Quick Reference

Se escribió una guía rápida en el README del proyecto, la cual puede ser visualizada en GitHub: https://github.com/A00820449/antlr4-final-project. Se decidió hacer esta guía en inglés, para que tuviera mayor universalidad.

Aquí se pone el código Markdown que se utilizó para esta guía:

```
# Final project for Compiler Design
* Miguel Angel Tornero Carrillo A00820449
* Github: <https://github.com/A00820449/antlr4-final-project>

## Image Manipulation Language (IML)
IML seeks to allow users to easily edit images using a simple,
C-like syntax. The image operations that are currently supported
are cropping, resizing, flipping vertically, flipping horizontally,
and rotating.

## How To Use
Make sure you have the latest version of
[NodeJS](https://nodejs.org/), [NPM](https://www.npmjs.com/)
```

(included with NodeJS), and [ANTLRv4](https://www.antlr.org/) installed. You can install ANTLRv4 via [PyPI](https://pypi.org/):
```bash
pip install antlr4-tools # other installation methods exist
```
After cloning the repository, install the NPM dependencies and build the grammar files using these commands:
```bash
npm install # installs NPM dependencies
npm run build # builds grammar files using ANTLR
```
Compiler
To use the compiler, use Node to run the file `compiler.js` like so:
```bash
node compiler.js [input_file_name] [output_file_name]
```
The first argument will be used as the name of the input file, otherwise it will default to `input.txt`.
The second argument will be used as the name of the output object file, otherwise it will default to `index.obj`.
Virtual Machine
To use the virtual machine, use Node to run the file `vm.js` like so:
```bash
node vm.js [input_file_name]
```
The first argument will be used as the name of the input object file, otherwise it will default to `index.obj`.
Program syntax
An IML program follows the following syntax
```
program programName:

[global variables]

[functions]

main:
{
    [statements]
}
```
As one can see, it's a very simple syntax.
The section indicated by `[global variables]` is where you declare the variables that will be used globally. You have to declare a variable here before you can use it in the main body. The section marked by `[functions]` indicates where you can declare your functions.
Variables
A variable declaration follows the following syntax:

```
var <type> <id> [, <id>];
```
A variable can have a `number` type or a `boolean` type. It can also be an array, or a double array. Here's an example of a declaration of a number variable, followed by the declaration of an array of 5 booleans, followed by a double array of numbers of size 5x9:
```
var number a;
var boolean[5] b;
var number[5][9] c;
```
Functions
A function declaration follows the following syntax:
```
function <type> functionName([parameters]):
[local variables]
{
    [statements]
}
```
A function can be of type `number`, `boolean`, or `void`. Parameters are declared in a comma separated list, in which you declare the type followed by the name of the Parameters. A local variable follows the same syntax as shown in the previous section. Here's an example of a function with the name `foo` that has a boolean parameter with name `a`, a number parameter of name `b`, has a local boolean variable with name `f`, and returns a boolean value:
```
function boolean foo(boolean a, number b):
var boolean f;
{
    return false;
}
```
Statements
As previously said, IML has a very C-like syntax, so a lot of the statements will result familiar to experienced programmers. Here's an example of what an if statement would look like:
```
{
    if (foo < barr) {
        print("Hello!");
    }
    else {
        print("Howdy!");
    }
}
```

This language comes with a lot of built-in functions to aid the programmer. Some of them include:
* `round(number)` Which rounds a decimal value to the nearest integer
* `trunc(number)` Which truncates a decimal value.
* `floor(number)` Which rounds a decimal value to the lowest integer
* `ceil(number)` Which rounds a decimal value to the highest integer
* `isInteger(number)` Which returns true if the number given is an integer.
<!-- end of the list -->
For image manipulation, we provide the following functions:
* `load(fileName)` which loads a file into memory
* `resize(w, h)` Which resizes the image to a width `w` and a height `h`
* `crop(x, y, w, h)` Which crops the image starting from the coordinates `x` and `y` to a width `w` and a height `h`
* `rotate(deg)` Which rotates the image a `deg` amount of degrees
* `flipHorizontally()` Which flips the image horizontally
* `flipVertically()` Which flips the image vertically
* `getWidth()` Which returns the current width of the loaded image
* `getHeight()` Which returns the current height of the loaded image
* `save(fileName)` Which saves the modified image to disk.
Example program
Putting all these things together, here's an example of a simple program:
```
program square:
var number h, w, m, i;
var boolean b;
function number min(number a, number b):
var number output;
{
    if (a < b) {
        output = a;
    }
    else {
        output = b;
    }
    return output;
}
function boolean not(boolean b):
{
    if (b) {
        return false;
    }
    return true;
}
main: {
```

```
    # this is a comment
    print("loading image...\n");
    load("test.png");
    w = getWidth();
    h = getHeight();

    m = min(w, h);

    # squaring image
    if (w != h) {
        print("cropping image...");
        crop(0,0,m,m);
    }
    else {
        print("image is already square");
    }

    save("output.png");

    # printing all even numbers from 1 to 100
    i = 1;
    b = false;
    while (i <= 100) {
        if (b) {
            print("even: ", i);
        }
        b = not(b);
        i = i + 1;
    }
}
```

```
## Demo Video
<a
href="http://www.youtube.com/watch?feature=player_embedded&v=uOECKF
EKpi4" target="_blank"><img
src="http://img.youtube.com/vi/uOECKFEKpi4/0.jpg" alt="Thumbnail
for Demo Video on YouTube" width="240" height="180" border="5"
/></a>
Link: <https://www.youtube.com/watch?v=uOECKFEKpi4>
```

Video demo

El video demo fue subido a YouTube, el link se comparte a continuación:

https://www.youtube.com/watch?v=uOECKFEKpi4

El video demo también está incluido en la Quick Reference y en el repositorio en GitHub.