

Proyecto final compiladores

Elda Guadalupe Quiroga González

Bernardo Orozco Garza A00819128

Erik Eduardo Velasco Gómez A00510780

Martes, Noviembre 20, 2018

Índice

Descripción del proyecto	4
Propósito	4
Objetivo	4
Alcance	4
Requerimientos y casos de uso	4
Test cases	5
Compilación	5
Ejecución	6
Proceso de desarrollo	6
Bitácora de cambios	7
Bitácora de desarrollo	7
Reflexión	8
Descripción del lenguaje	8
Nombre del lenguaje	8
Descripción del lenguaje	8
Errores que pueden ocurrir	9
Compilación	9
Ejecución	9
Descripción del compilador	9
Equipo de cómputo, lenguaje y utilerías especiales usadas en el desarrollo del proyecto.	9
Descripción del Análisis de Léxico.	9
Patrones de construcción.	9
Enumeración de tokens del lenguaje y su código asociado.	10
Descripción del Análisis de Sintaxis	11
Gramática Formal empleada para representar las estructuras sintácticas	11
Descripción de Generación de Código Intermedio y Análisis Semántico	11
Diagramas de Sintaxis con las acciones correspondientes	11
Descripción de las acciones sintácticas	15
Breve descripción de código de operación y dirección virtual asociada	17
Descripción del proceso de Administración de Memoria usado en la compilación.	19
Especificación gráfica de CADA estructura de datos usada	19
Descripción de la máquina virtual	20
Equipo de cómputo, lenguaje y utilerías especiales usadas en el desarrollo del proyecto.	20

Especificación gráfica de CADA estructura de datos usada para manejo de scopes	21
Asociación hecha entre las direcciones virtuales (compilación) y las reales (ejecución).	22
Pruebas del funcionamiento del lenguaje	22
Factorial lineal	22
Factorial recursivo	23
Find lineal	23
Find recursivo	25
Sort lineal	26
Sort recursivo	28
Dibujar objetos	30
Listados documentados del proyecto	34
main.py	34
Visitor.py	34
quadruplets.py	35
algorithm_with	35
token_to_dir	35
gen_array_access	37
function_dir.py	38
create_function_vars	38
add_variable	39
add_function	39
custom_stack.py	40
pop_arg	40
top_arg	40
exe_memory.py	41
get	41
add	41

Descripción del proyecto

Propósito

Lo que se quiere lograr a hacer con este proyecto, es llevar la programación a todas las personas que hablen español, para ayudarlos a aprender a programar en su lenguaje, nos gustaría llegar a mostrar este proyecto y que lo utilicen en el municipio de Monclova, COA, ya que no se conoce mucho de programación y tampoco enseñan muy bien inglés, así podrían empezar a temprana edad sin el requisito de aprender inglés.

Objetivo

Poder tener un lenguaje de programación, funcionando que esté completamente en español para niños de secundaria o preparatoria, que estén interesados en programar, pero no saben inglés, así que quieren seguir aprendiendo se darán la oportunidad de estudiar inglés y seguir con lenguajes de programación más avanzados.

Alcance

Llegar a tener un lenguaje de programación, con variables simples como enteros y flotantes, también arreglos, poder ejecutar código no lineal por medio de condiciones y ciclos, si es posible llegar a tener funciones recursivas para dar una introducción a la programación básica con Daxe.

Requerimientos y casos de uso

Id	Descripción
RF1	El usuario podrá ejecutar desde un archivo con extensión .dax
RF2	El usuario podrá ejecutar por medio de un archivo binario su programa
RF3	El usuario podrá declarar variables de tipo entero, flotante y arreglo de dichos tipos
RF4	El usuario podrá declarar sus propias funciones que regresen tipos void, entero y flotante
RF5	El usuario deberá poder generar recursión en las funciones
RF6	Los usuarios podrán generar triángulos, círculos, cuadrados y dibujar texto en la pantalla
RF7	El usuario podrá mover libremente un lápiz en la pantalla por medio de las funciones de adelante y rotar.
RF8	Al finalizar el programa el usuario podrá ver su obra en un archivo llamado image.svg

Id	CU1
Actores	Usuario
Descripción	El programa marcará type mismatch
Pre condición	Tener una variable de tipo entero declarada
Flujo principal	<ol style="list-style-type: none">1. A la variable declarada asignar un flotante2. En la consola desplegará 'Type mismatch trying to assign (type: decimal) to &i (type: entero), at: %s:%s'

Flujo alternativo	
Post condición	

Id	CU2
Actores	Usuario
Descripción	El programa deberá imprimir syntax error
Pre condición	
Flujo principal	<ol style="list-style-type: none"> 1. Tener una variable declarada 2. Al momento de asignar, terminar con punto y coma, sin contenido en medio 3. La consola debe de imprimir 'No terminal defined for ';' at line %s col %s'
Flujo alternativo	
Post condición	

Id	CU3
Actores	Usuario
Descripción	El programa deberá marcar uninitialized variable
Pre condición	
Flujo principal	<ol style="list-style-type: none"> 1. Declarar una variable 2. La variable no se le asigna un valor 3. Utilizar la variable en una condición, ciclo, función, expresión. 4. En la consola se mostrará 'Uninitialized variable'
Flujo alternativo	
Post condición	

Test cases

Compilación

ID	Descripción	código	Resultado
TCC1	No terminar estatuto con punto y coma	<code>&i = 120</code>	X
TCC2	Utilizar una variable no declarada	<code>&NO = 10;</code>	X
TCC3	Utilizar una función no declarada	<code>~noDeclarada(10);</code>	X
TCC4	Pasar parametro diferente tipo	<code>imprimir ~factorial(10.5);</code>	X
TCC5	No declarar función main	<code>programa "uno";</code>	X
TCC6	Ciclo sin paréntesis final	<code>&i=0; mientras(&i < 5){</code>	X

		<pre> leer &lista[&i]; &i=&i+1; </pre>	
TCC7	Condicional sin paréntesis final	<pre> si(&index > -1){ imprimir &index; imprimir &lista[&index]; }sino{ imprimir &index; } </pre>	X

Ejecución

ID	Descripción	código	Resultado
TCE1	Leer variable entera (input entero)	leer &i;	O
TCE2	Leer variable entera (input decimal)	leer &i;	X
TCE3	Leer variable decimal (input entero)	leer &f;	X
TCE4	Leer variable decimal (input decimal)	leer &f;	O
TCE5	Asignar entero un entero		
TCE6	Asignar entero un decimal		
TCE7	Asignar decimal un entero		
TCE8	Asignar decimal un decimal		
TCE9	Mover lápiz entero		
TCE10	Mover lápiz decimal		
TCE11	Rotar entero		
TCE12	Rotar decimal		

Proceso de desarrollo

Al principio se realizó una investigación intensiva de diferentes lenguajes de programación, las mejores librerías para parsers sintácticos, las tendencias en la programación, se investigó en librerías para poder crear los diagramas de sintaxis de manera gráfica y programable para poder hacer cambios rápidos en las bitácoras.

Para los registros de cambios utilizamos el control de versiones con la herramienta de github, para tener récord de cambios ejecutados en la documentación y el compilador, esto se realiza por medio de las nuevas publicaciones en el repositorio.

Cada semana se planteaba una meta y pruebas incrementales (conforme a los entregables descritos en el calendario), ya que en cada integración se probaba exhaustivamente, pero con el mismo archivo así que al final el archivo contiene todas las pruebas ejecutadas.

Bitácora de cambios

Fecha	Descripción
22-Sep-2018	Solucionar aceptar expresiones en los parámetros en lugar de valores predefinidos (dibujar objetos)
10-Oct-2018	Cambio en token de numero int y flotante (aceptar directamente signo negativo opcional)
26-Oct-2018	Aceptar más de un estatuto en el cuerpo de las funciones
15-Nov-2018	Agregar estatuto de lectura

Bitácora de desarrollo

Fecha	Descripción
22-sep-2018	Setup del proyecto y gramática inicial
6-oct-2018	Creación de directorio de funciones, visitador bottom up, creación de cubo semántico
9-oct-2018	Creación de directorio de variables.
10-oct-2018	Creación de clase custom_stack para operadores y operandos, integración de estatutos de condición y ciclos, actualización del cubo semántico.
11-oct-2018	Generación de cuádruplos para ciclos, detección de declaraciones múltiples, generacion de quadriplos para asignación.
21-oct-2018	Acciones para preparar llamada a función, agregar funcionalidad para poder dibujar objetos y mover lápiz libremente.
26-oct-2018	Cambio de nombres de variables por direcciones, inicialización de la máquina virtual

27-oct-2018	Creación de la memoria de ejecución, aceptación de recursión.
3-Nov-2018	Funcionalidad de exportación a SVG, crear restricción de stack overflow.
4-Nov-2018	Puntos neurálgicos para arreglos, directorio de variables agregar el campo de dimensión, creación de binario para ejecución.
15-Nov-2018	Agregar funcionalidad de lectura para variables, soporte de comentarios (* *) y //

Reflexión

Bernardo Orozco: Nunca se había trabajado con python en un proyecto solo se tenía experiencia en desarrollo de scripts básicos, pero este ayudó a descubrir el potencial de este lenguaje de programación y lo sencillo que es para poder hacer cosas muy complejas gracias a la comunidad y los proyectos libres con los que cuenta. La parte más difícil en mi punto de vista fue las funciones recursivas, por que no entendía que estaba pasando qué parámetros se estaban pasando o por que regresaba un número erróneo cada vez que se corría, hasta que descubrí que los valores de los parámetros no se estaban inicializando.

Erik Velasco: Creo que fue altamente interesante haber desarrollado este compilador, pues con ello se pudieron reafirmar múltiples conceptos vistos en clase, pasando desde el análisis léxico, el sintáctico, verificando las validaciones semánticas y generando el código intermedio, llegando hasta la creación de la máquina virtual. Habernos ayudado del lenguaje de programación python y de sus librerías lark, y turtle facilitó mucho las cosas pues pudimos comprobar el gran poder del lenguaje y de todas estas herramientas, la facilidad de operaciones de manipulación de estructuras de datos y de utilización de funciones es increíble y no hubiera sido posible con otros lenguajes de alto nivel como C++ o Java.

Descripción del lenguaje

Nombre del lenguaje

Daxe

Descripción del lenguaje

Lenguaje de programación orientado a la creación de imágenes digitales por medio de código, los elementos que ofrece para dibujar es un cuadrado, círculo, triángulo, etiqueta de texto, además de tener un lápiz que se puede mover libremente, al terminar la ejecución del programa la pantalla se quedará congelada y creará un archivo SVG donde demuestra lo que has creado en con líneas de código, dicho lenguaje está basado en el español, con la intención de promover la programación en secundarias y prepas donde no tengan conocimiento del lenguaje inglés.

Errores que pueden ocurrir

Compilación

- Declaración múltiple de funciones con el mismo nombre %s
- Declaración múltiple de variable(%s) en la función(%s)
- Función %s indefinida en %s:%s
- No proporcionaste archivo de entrada
- Tipos no coinciden en la asignación (tipos: %s) a %s (tipos: %s), en: %s:%s
- Tipos no coinciden tratando de evaluar si en %s:%s
- Función no declarada con el mismo tamaño de parámetros en %s:%s
- Variable (%s) no es un arreglo en %s:%s
- Variable no definida %s, en: %s:%s
- Tipos no coinciden con el regreso (esperado: %s) (dado: %s) en %s:%s
- El índice del arreglo (%s) no es un entero en %s:%s
- Tipos no coinciden en la asignación (tipo: %s) al parámetro %s (tipo: %s), en: %s:%s
- Fin inesperado de input! Esperando terminal de tipo: %s

Ejecución

- Variable no inicializada
- Clave no reconocida: %s
- Pila sobrecargada llamando a la función %s
- Pila sobrecargada
- Acción no reconocida %s
- Índice del arreglo fuera de rango

Descripción del compilador

Equipo de cómputo, lenguaje y utilerías especiales usadas en el desarrollo del proyecto.

Equipo de cómputo: MacOS

Lenguaje: Python 2.7

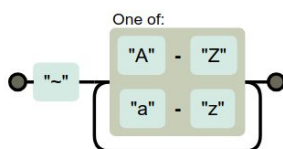
Utilerías especiales: LARK.py

Descripción del Análisis de Léxico.

Patrones de construcción.

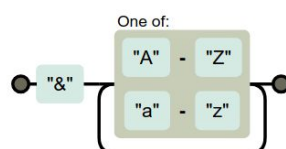
T_FUN_ID:

$\wedge^*[A-Za-z]^+/g$



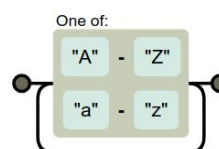
T_VAR_ID:

$/\&[A-Za-z]^+/g$



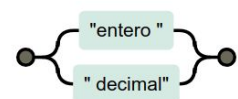
T_ID:

$/[A-Za-z]^+/g$



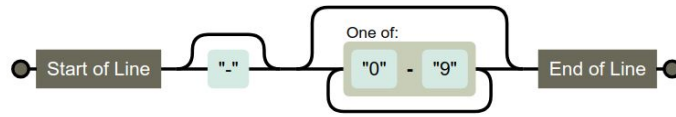
T_VAR_TYPE:

$/\text{entero} \mid \text{decimal}/g$



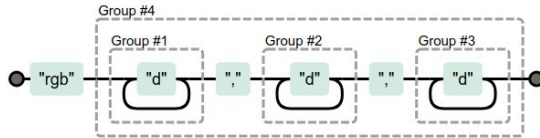
T_NUM_INT:

/^-?[0-9]*\$/gm



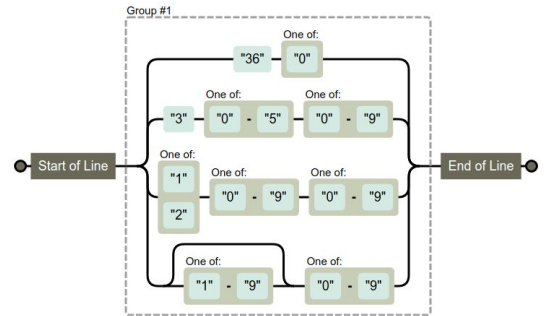
T_COLOR:

/rgb\((\d+),(\d+),(\d+)\)/g



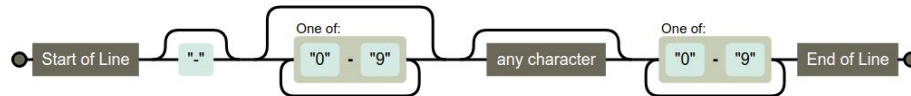
T_NUM_DEG:

^\^(36[0]3[0-5][0-9][12][0-9][0-9][1-9]?[0-9])\$/gm



T_NUM_FLOAT:

/^-?[0-9]*\.[0-9]+\$/gm



Enumeración de tokens del lenguaje y su código asociado.

- | | |
|-------------------------------------------|------------------------------|
| 1. T_PROGRAM: "programa" | 26. T_CUADRADO: "cuadrado" |
| 2. T_COMILLA: /(\\" \\')/ | 27. T_TRIANGULO: "triangulo" |
| 3. T_ID: /[A-Za-z0-9!@#\$%^&*(,.;:} <>]+/ | 28. T_TEXTO: "texto" |
| 4. T_PUNTO_COMA: ";;" | 29. T_IGUAL: "=" |
| 5. T_VAR: "var" | 30. T_POUND: "#" |
| 6. T_VAR_ID: /[&[A-Za-z]+/ | 31. T_PLUS: "+" |
| 7. T_PUNTO_PUNTO: ":" | 32. T_MINUS: "-" |
| 8. T_VAR_TYPE: /entero decimal/ | 33. T_DIVITION: "/" |
| 9. T_LEFT_BRACKET: "[" | 34. T_MULTIPLICATION: "*" |
| 10. T_RIGHT_BRACKET: "]" | 35. T_IF: "si" |
| 11. T_NUM_INT: SIGNED_INT | 36. T_ELSE: "sino" |
| 12. T_COMMA: "," | 37. T_IMPRIMIR: "imprimir" |
| 13. T_FUN: "funcion" | 38. T_MAYOR_QUE: ">" |
| 14. T_VOID: "void" | 39. T_MENOR_QUE: "<" |
| 15. T_FUN_ID: /[~[A-Za-z]+/ | 40. T_IGUAL_IGUAL_QUE: "==" |

16. T_LEFT_PAR: "("
 17. T_RIGHT_PAR: ")"
 18. T_LEFT_CRULY_BRACKET: "{"
 19. T_RIGHT_CRULY_BRACKET: "}"
 20. T_RETURN: "regresar"
 21. T_DIBUJAR: "dibujar"
 22. T_ADELANTE: "adelante"
 23. T_ROTAR: "rotar"
 24. T_MIENTRAS: "mientras"
 25. T_CIRCULO: "circulo"

41. T_MAYOR_IGUAL_QUE: ">="
 42. T_MENOR_IGUAL_QUE: "<="
 43. T_DIFERENTE_QUE: "<>"
 44. T_COLOR:
 /rgb\\(\\s*(?:\\d{1,2}|1\\d\\d|2(?:[0-4]\\d|5[0-5]))\\s*,?\\s*\\s*{3}\\s*)\\s*\\/

45. T_NUM_DEG:
 /(36[0]|3[0-5][0-9]|[12][0-9][0-9]|[1-9]?[0-9])/

46. T_NUM_FLOAT: SIGNED_FLOAT

Descripción del Análisis de Sintaxis

Gramática Formal empleada para representar las estructuras sintácticas

Se utilizó una gramática libre de contexto, se utilizó la versión Earley que te permite parsear gramáticas ambiguas sin problemas, la se podría definir como si tuviera recursión por la izquierda pero la librería utilizada lo permite utilizar y lo parsea sin problema y dicha librería contiene una función de loop en recursión por la derecha bottom up para generar acciones por cada nodo generado.

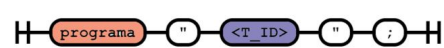
Descripción de Generación de Código Intermedio y Análisis Semántico

Diagramas de Sintaxis con las acciones correspondientes

INICIAR_PROGRAMA



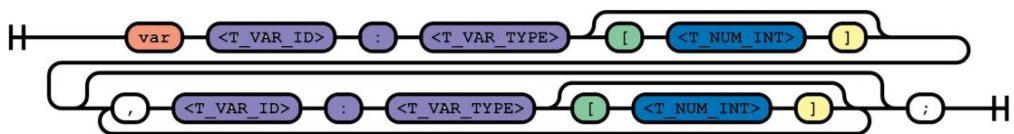
NOMBRE_PROGRAMA



a_t_programa

a_t_id_programa

VARIABLES



a_t_var

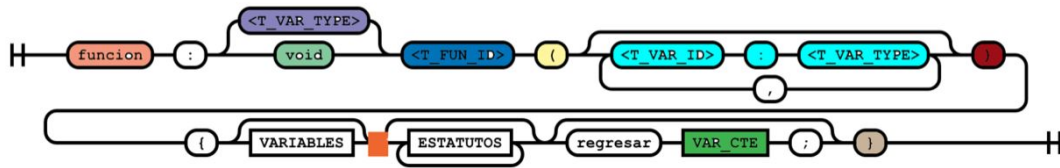
a_g_var_array_left

a_t_var_id_y_tipo

a_g_var_array_size

a_g_var_array_right

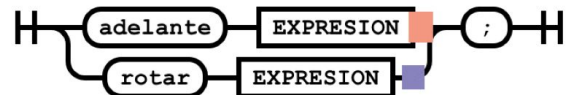
FUNCIONES



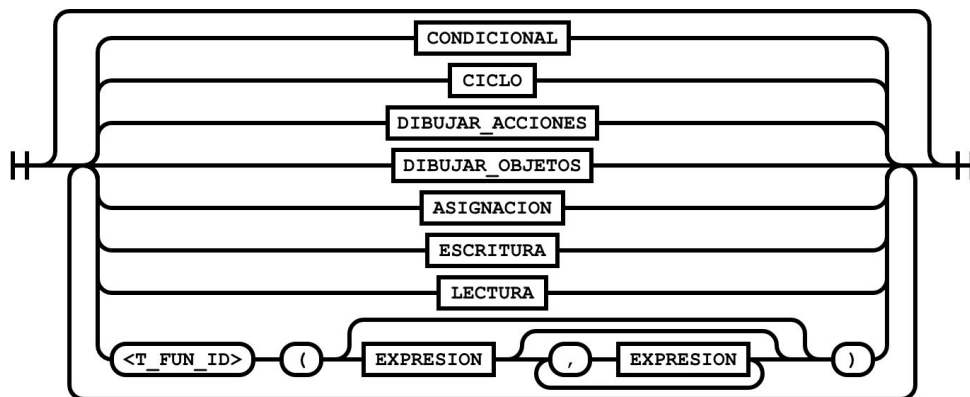
MAIN



DIBUJAR_ACCIONES



ESTATUTOS



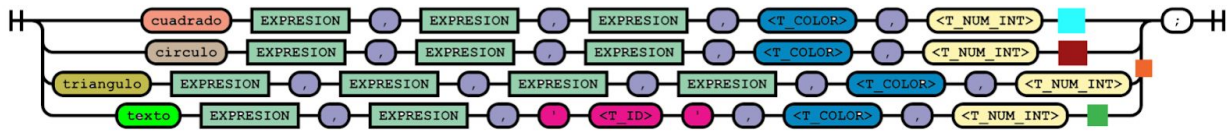
DIBUJAR_OBJETOS

cuadrado x, y, lado, #hex_color(relleno), grosor_linea

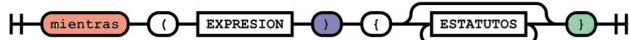
circulo x, y, radio, #hex_color(relleno), grosor_linea

triangulo x,y,base,altura, #hex_color(relleno), grosor_linea

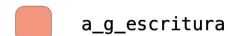
texto x, y, texto, #hex_color, tamaño letra



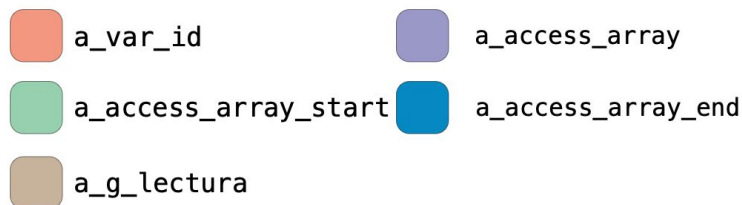
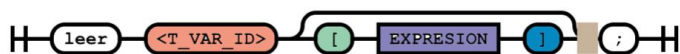
CICLO



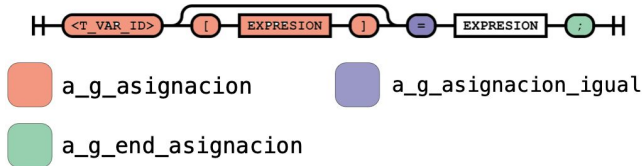
ESCRITURA



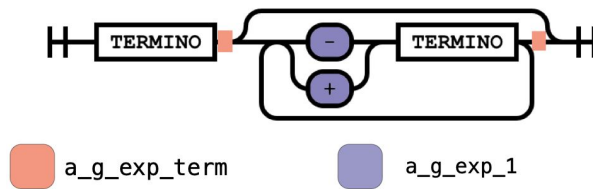
LECTURA



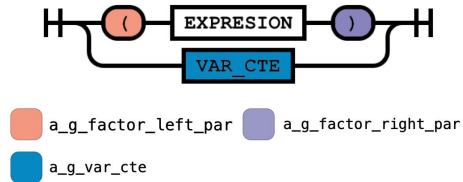
ASIGNACION



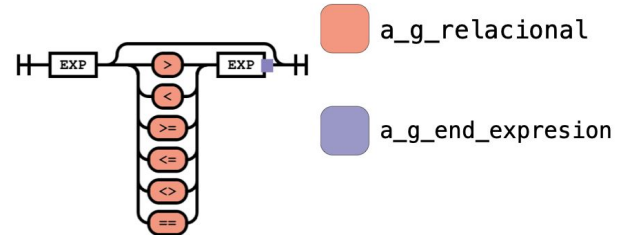
EXP



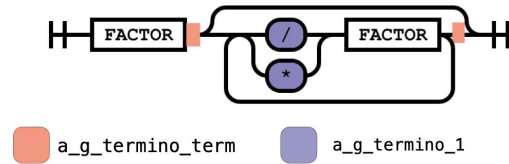
FACTOR



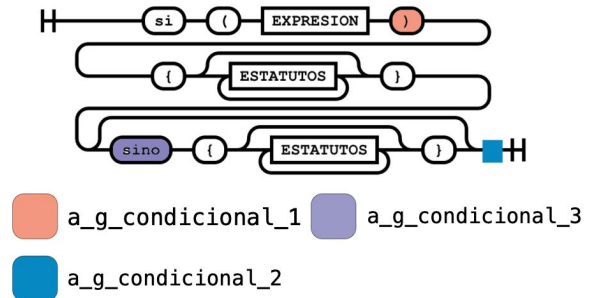
EXPRESION



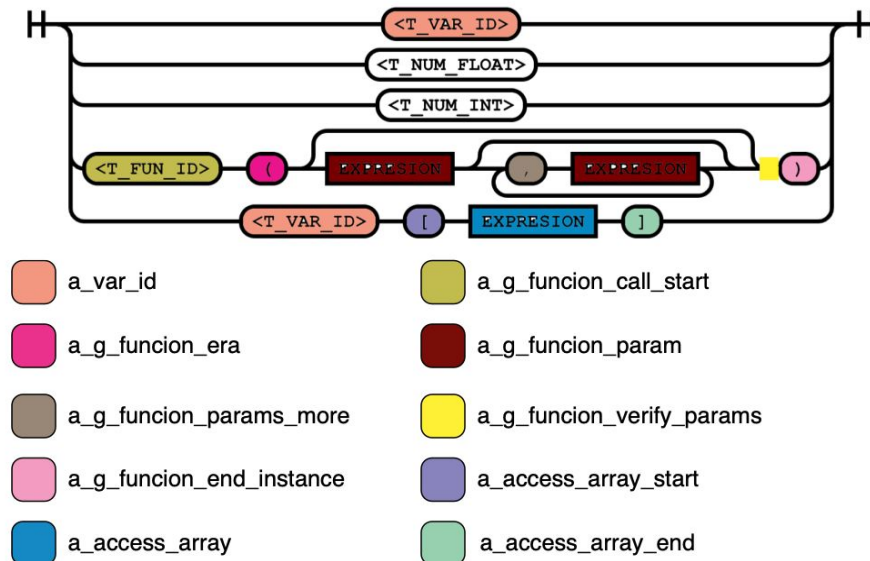
TERMINO



CONDICIONAL



VAR_CTE



Descripción de las acciones sintácticas

Id	Descripción
a_t_programa	crear directorio de funciones y su directorio de variables
a_t_id_programa	crear la primera función
a_t_var	crear una tabla de variables en la función actual
a_t_var_id_y_tipo	agregar variable con el tipo actual y id
a_t_var_type	asignar el tipo a actual
a_t_end_program	término del programa, eliminar directorio de funciones e inicializar la máquina virtual
a_t_var_void	agregar el tipo actual a void
a_t_fun_id	agregar una nueva función a la tabla de funciones
a_t_fun_l_par	crear tabla de variables
a_t_fun_r_par	vaciar la columna de parámetros
a_g_fun_start_exec	agregar la semilla a regresar cuando termine la función
a_g_return	generar cuadruplo de regresar
a_t_end_function	crear quad de endproc, reiniciar contadores de variables y eliminar tabla de variables
a_g_end_expresion	evaluar algoritmo de quads con ">", "<", "==", ">=", "<=", "<>"
a_g_relacional	agrega a la pila de operadores
a_g_exp_1	agregar a la pila de operadores + o -
a_g_exp_term	evaluar algoritmo de quads con "-", "+"
a_g_termino_1	agregar a la pila / o *
a_g_termino_term	evaluar algoritmo de quads con "/", "**"
a_g_factor_left_par	agregar fondo falso a la pila de operadores
a_g_factor_right_par	pop de la pila de operadores
a_g_var_cte	agregar id a la pila de operandos con su tipo

a_g_asignacion	agregar operando de la zona izquierda del operando
a_g_asignacion_igual	agregar a la pila =
a_g_end_asignacion	evaluar algoritmo de quads con "="
a_g_escritura	generar quad de print sacando el último operando de la pila
a_g_lectura	genera quad de read sacando de la pila el último operador para saber qué variable
a_g_dibujar_rotar	genera cuádruplo de rotar con el último operando
a_g_dibujar_adelante	generar cuádruplo de mover lápiz adelante con el último operando
a_g_condicional_1	evaluar tipo booleano y su último operando para poder generar el gotof, guardar pila de saltos
a_g_condicional_2	sacar de la pila de saltos y rellenar los vacíos
a_g_condicional_3	generación de cuádruplos para el else
a_g_ciclo_start	agregar cuádruplo a la pila de jumps
a_g_ciclo_mid	generar gotof después de evaluar expresión
a_g_ciclo_end	generar goto a evaluar la expresión, fill del gotof
a_g_funcion_call_start	agregar fondo falso a los operadores y agregar a la pila de funciones el nombre
a_g_funcion_era	generar el cuádruplo de era e inicializa los parámetros counter
a_g_funcion_param	pop de operandos y tipos y validar posición y el tipo que tenga.
a_g_funcion_params_more	sumar uno al contador de parámetros
a_g_funcion_verify_params	validar que tengan el mismo tamaño
a_g_funcion_end_instance	generar el gosub de la función en caso de return genera un asignacion después del gosub
a_g_cuadrado_init	generar era para el cuadrado 'SCUAD'
a_g_cuadrado_end	genera gosub para el cuadrado 'ECUAD'
a_g_circulo_init	generar era para el círculo 'SCIR'

a_g_circulo_end	genera gosub para el círculo 'ECIR'
a_g_triangulo_init	generar era para el triángulo 'STRI'
a_g_triangulo_end	genera gosub para el triángulo 'ETRI'
a_g_texto_init	generar era para el texto 'STXT'
a_g_texto_end	genera gosub para el texto 'ETXT'
a_g_exp_param	pop pila de operadores para poder insertar en parámetros
a_g_draw_p_one	incrementar el contador de parámetros
a_g_draw_t_color	pasar rgb como parámetros y generar quad
a_g_draw_stroke	generar parámetro de grosor de contorno
a_g_draw_txt_body	similar a a_g_draw_t_color pero con texto
a_g_main	rellenar el primer goto
a_g_var_array_left	validar si la variable es un arreglo
a_g_var_array_size	crear la columna en la tabla de variable y almacenar su tamaño
a_g_var_array_right	asignar la dirección de base y sumarle al contador de direcciones el tamaño del arreglo
a_var_id	push a la pila de operandos el id y el tipo para variables
a_access_array_start	verificar que es un arreglo la variable y asignar un fondo falso para evaluar expresión interna
a_access_array	genera el quad de verificación con límite superior
a_access_array_end	generar el cuádruplo de dirección el tipo '(dir)'

Breve descripción de código de operación y dirección virtual asociada

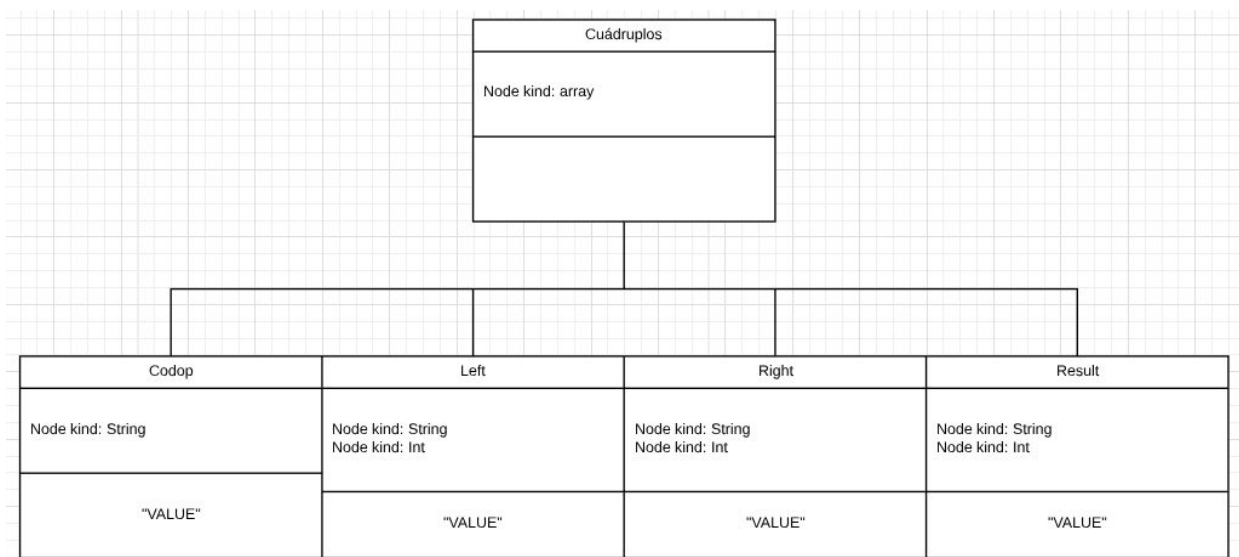
Id	Código	Descripción
0	+	Suma de elementos de cuádruplo
1	-	Resta de elementos de cuádruplo

2	*	Multiplicación de elementos de cuádruplo
3	/	División de elementos de cuádruplo
4	==	Comparación de igualdad de elementos de cuádruplo
5	>=	Comparación de “mayor o igual que” de elementos de cuádruplo
6	<=	Comparación de “menor o igual que” de elementos de cuádruplo
7	>	Comparación de “mayor que” de elementos de cuádruplo
8	<	Comparación de “menor que” de elementos de cuádruplo
9	<>	Comparación de “diferente que” de elementos de cuádruplo
10	=	Asignación de elementos de cuádruplo
11	GOTO	Salto a otro cuádruplo
12	GOTOF	Salto en falso a otro cuádruplo
13	GOTOV	Salto en verdadero a otro cuádruplo
14	PRINT	Despliegue en pantalla de elemento de cuádruplo
15	MOVF	Mueve el puntero del lápiz en puntos de coordenadas
16	ROT	Rota la cabeza del lápiz hacia la izquierda en grados
17	PARAM	Asignación de elementos de cuádruplo a parámetro de función
18	GOSUB	Salto a la función y asigna los parámetros leídos con anterioridad
19	SCUAD	Empezar a dibujar el cuadrado, agregar fondo falso
20	ECUAD	Saca todos los parámetros para poder crear el cuadrado
21	SCIR	Empezar a dibujar el círculo, agregar fondo falso
22	ECIR	Sacar todos los parámetros almacenados para poder dibujar círculo
23	STRI	Empezar a almacenar parámetros para triángulo, insertar fondo falso
24	ETRI	Sacar los parámetros y poder customizar el triángulo
25	STXT	Empezar a almacenar parámetros en los textos
26	ETXT	Sacar los parámetros del texto, para poder crearlo.

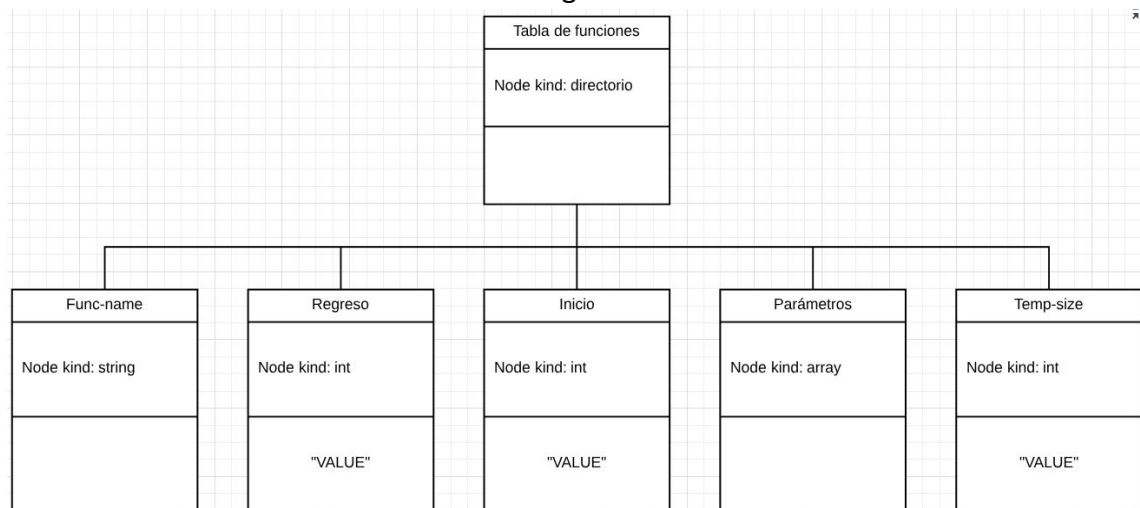
27	ENDPROC	Marca el final de cada función (módulo) regresa al punto anterior
28	ERA	Separa la cantidad de memoria a utilizar y crea fondo falso
29	RETURN	Regresa un valor de una función
30	READ	Genera la lectura de una variable y se la asigna

Descripción del proceso de Administración de Memoria usado en la compilación.

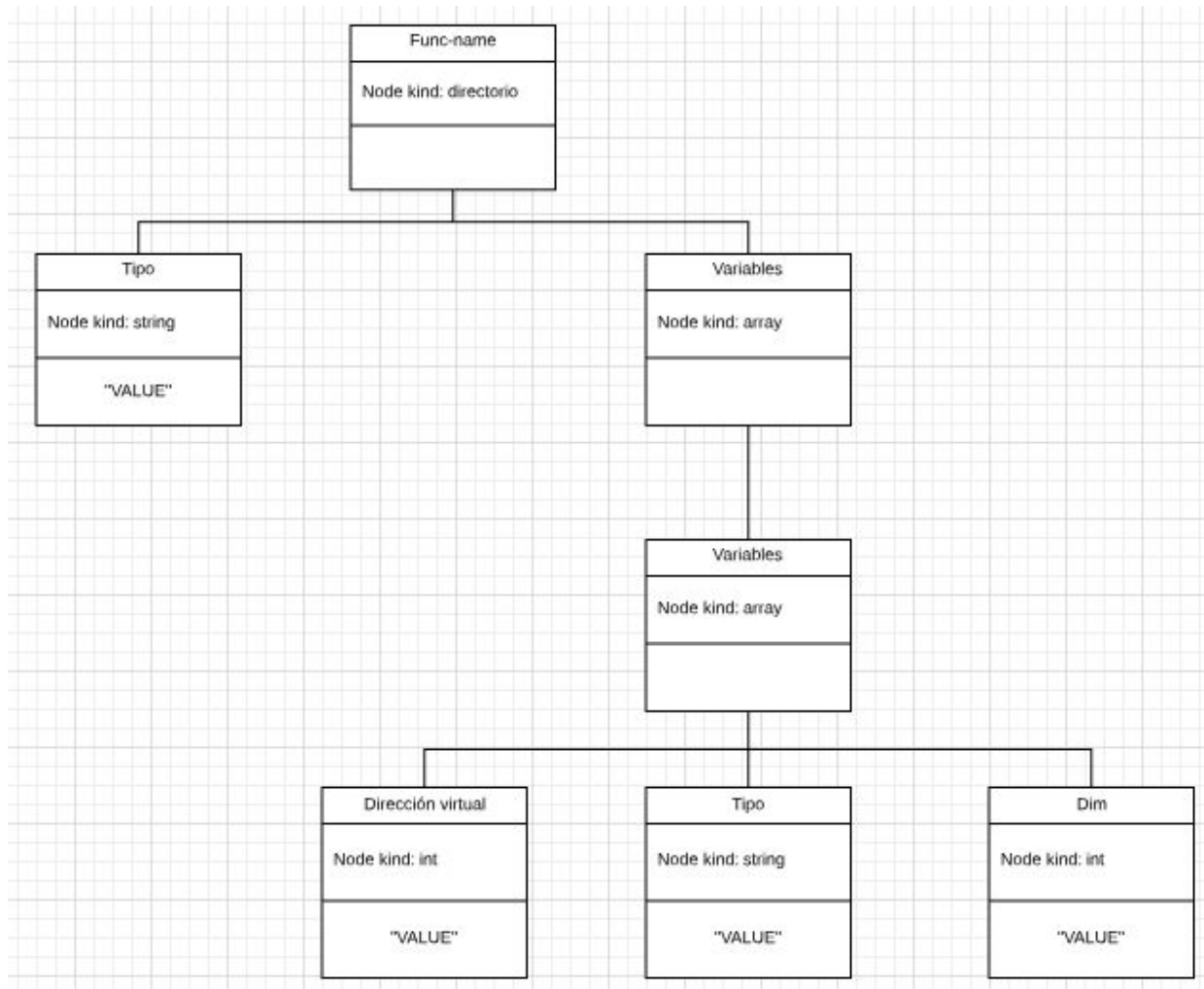
Especificación gráfica de CADA estructura de datos usada



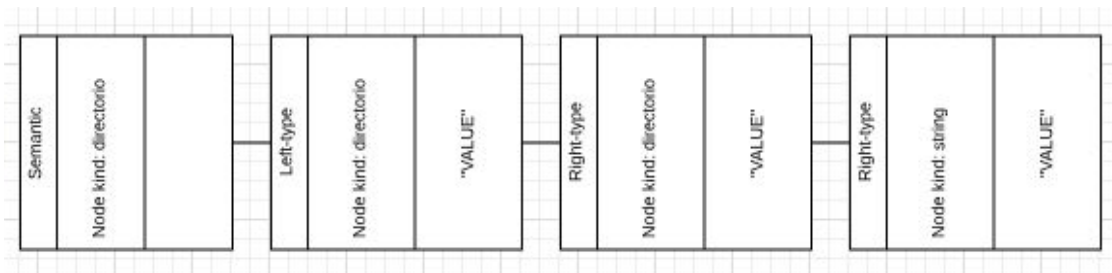
Vector de arreglos de 4 valores



Directorio de directorios internos para variables



Propiedades adicionales a los directorios hijos de la tabla de funciones



Directorio de directorios(4 niveles)

Descripción de la máquina virtual

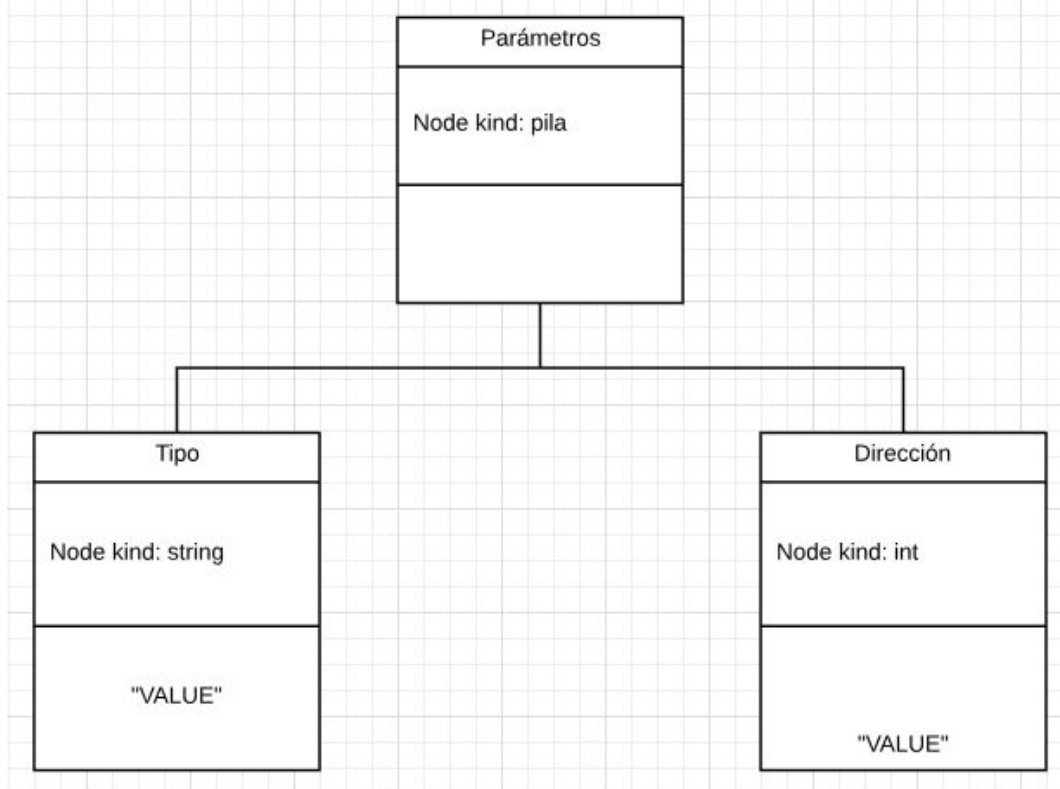
Equipo de cómputo, lenguaje y utilerías especiales usadas en el desarrollo del proyecto.

Equipo de cómputo: MacOS

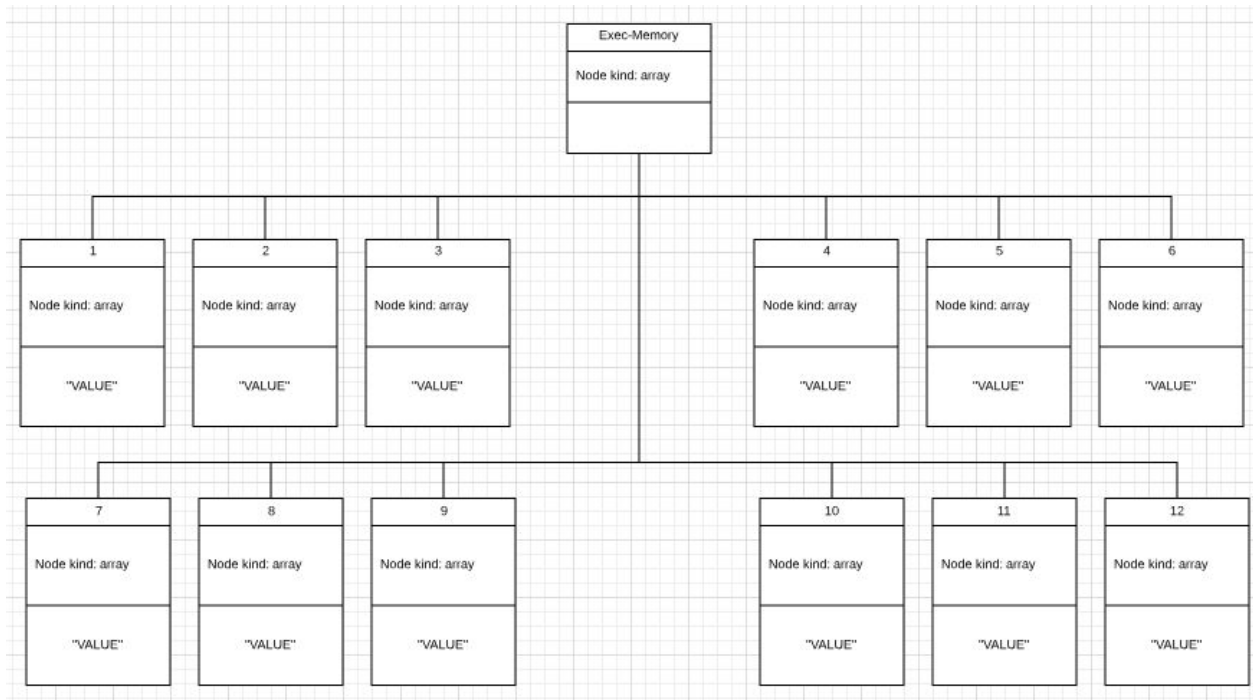
Lenguaje: Python 2.7

Utilerías especiales: turtle.py (para poder dibujar en las pantallas), pyinstaller (crear binario)

Especificación gráfica de CADA estructura de datos usada para manejo de scopes



Pila de pilas para llamadas recursivas



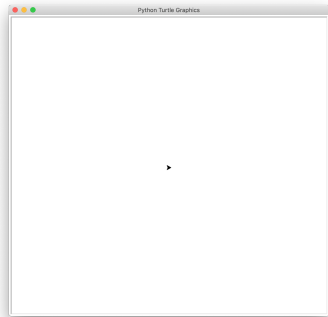
Directorio de vectores

Asociación hecha entre las direcciones virtuales (compilación) y las reales (ejecución).
 Para poder asociar las direcciones virtuales y reales en compilación se implementan contadores para cada dirección y entonces se mapean a direcciones reales en ejecución.
 Eg. Dirección virtual 2010 -> Posición 2 para memoria de ejecución, posición 10 de la lista.

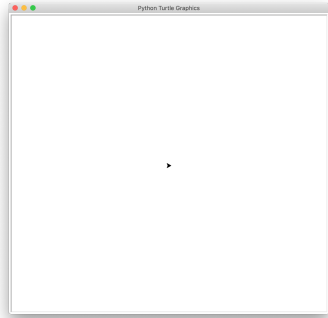
Internamente en la fase de compilación, la clase de tabla de funciones se tenía los contadores para las direcciones globales y globales temporales para los parámetros y las variables de retorno. En cuanto a las variables temporales se tenía el contador en la clase de cuádruplos para poder tener un contadores temporal de los enteros, booleanos, constantes y decimales (las direcciones temporales se reinician en cada fin de función).

Pruebas del funcionamiento del lenguaje

Factorial lineal

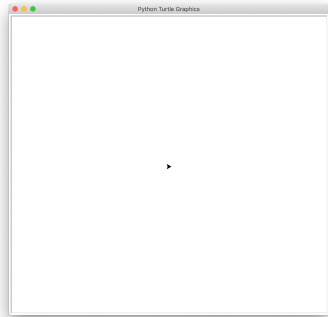
Código	Cuádruplos
<pre> programa "factorial"; dibujar(){ var &i : entero, &n : entero, &factorial : entero; leer &n; &factorial = 1; &i = 1; mientras(&i <= &n){ &factorial = &factorial*&i; &i=&i+1; } imprimir &factorial; } </pre>	<pre> ['GOTO', None, None, 1], ['READ', None, None, 8001], ['=', 5000, None, 8002], ['=', 5000, None, 8000], ['<=', 8000, 8001, 12000], ['GOTO', 12000, None, 11], ['*', 8002, 8000, 10002], ['=', 10002, None, 8002], ['+', 8000, 5000, 10004], ['=', 10004, None, 8000], ['GOTO', None, None, 4], ['PRINT', None, None, 8002] </pre>
Consola	Imagen
<pre> leyendo:6 720 </pre>	

Factorial recursivo

Código	Cuádruplos
<pre> programa "factorial"; funcion : entero ~factorial(&i : entero){ var &tmp : entero; si(&i >= 1){ &tmp = &i*~factorial(&i-1); }sino{ &tmp = 1; } regresar &tmp; } dibujar(){ var &n : entero; leer &n; imprimir ~factorial(&n); } </pre>	<pre> ['GOTO', None, None, 14], ['>=', 8000, 5000, 12000], ['GOTO', 12000, None, 11], ['ERA', None, None, '~factorial'], ['-', 8000, 5000, 10000], ['PARAM', 10000, None, 'param0'], ['GOSUB', None, None, '~factorial'], ['=', 1000, None, 10001], ['*', 8000, 10001, 10002], ['=', 10002, None, 8001], ['GOTO', None, None, 12], ['=', 5000, None, 8001], ['RETURN', None, None, 8001], ['ENDPROC', None, None, None], ['READ', None, None, 8000], ['ERA', None, None, '~factorial'], ['PARAM', 8000, None, 'param0'], ['GOSUB', None, None, '~factorial'], ['=', 1000, None, 10000], ['PRINT', None, None, 10000] </pre>
Consola	Imagen
<pre> leyendo:5 120 </pre>	

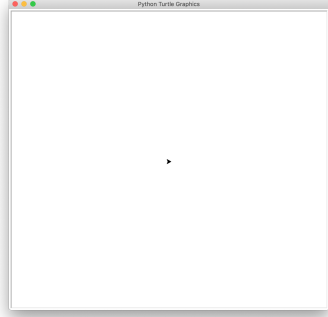
Find lineal

Código	Cuádruplos
<pre> programa "find"; dibujar(){ </pre>	<pre> ['GOTO', None, None, 1], ['=', 5000, None, 8000], </pre>

<pre> var &i : entero, &lista : entero[100], &encontrar : entero, &index : entero, &n : entero; &i=0; &index=-1; leer &n; mientras(&i < &n){ leer &lista[&i]; &i=&i+1; } leer &encontrar; &i=0; mientras(&i < &n){ si(&lista[&i] == &encontrar){ &index = &i; &i = 5; <i>//simular break;</i> } &i=&i+1; } si(&index > -1){ imprimir &index; imprimir &lista[&index]; }sino{ imprimir &index; } } </pre>	<pre> ['=', 5001, None, 8102], ['READ', None, None, 8103], ['<', 8000, 8103, 12000], ['GOTO', 12000, None, 12], ['VERIFY', 8000, 0, 100], ['+', 8000, 5002, 10002], ['READ', None, None, '(10002)'], ['+', 8000, 5003, 10003], ['=', 10003, None, 8000], ['GOTO', None, None, 4], ['READ', None, None, 8101], ['=', 5000, None, 8000], ['<', 8000, 8103, 12001], ['GOTO', 12001, None, 25], ['VERIFY', 8000, 0, 100], ['+', 8000, 5002, 10006], ['==', '(10006)', 8101, 12002], ['GOTO', 12002, None, 22], ['=', 8000, None, 8102], ['=', 5004, None, 8000], ['+', 8000, 5003, 10009], ['=', 10009, None, 8000], ['GOTO', None, None, 14], ['>', 8102, 5001, 12003], ['GOTO', 12003, None, 32], ['PRINT', None, None, 8102], ['VERIFY', 8102, 0, 100], ['+', 8102, 5002, 10011], ['PRINT', None, None, '(10011)'], ['GOTO', None, None, 33], ['PRINT', None, None, 8102] </pre>
Consola	Imagen
<pre> leyendo:4 leyendo:1 leyendo:2 leyendo:3 leyendo:4 leyendo:3 2 3 </pre>	

Find recursivo

Código	Cuádruplos
<pre> programa "findRecursive"; var &lista : decimal[100]; funcion : entero ~find(&izq : entero, &der : entero, &ele : entero){ var &tmp : entero; si(&izq > &der){ &tmp = -1; }sino{ si(&lista[&izq] == &ele){ &tmp = &izq; }sino{ si(&lista[&der] == &ele){ &tmp = &der; }sino{ &tmp = ~find(&izq + 1, &der - 1, &ele); } } } regresar &tmp; } dibujar(){ var &n : entero, &i : entero, &index : entero, &enc : decimal; leer &n; &i=0; mientras(&i<&n){ leer &lista[&i]; &i=&i+1; } leer &enc; &index = ~find(0, &n-1, &enc); si(&index > -1){ imprimir &index; imprimir &lista[&index]; }sino{ </pre>	<pre> ['GOTO', None, None, 28], ['>', 8000, 8001, 12000], ['GOTO', 12000, None, 5], ['=', 5000, None, 8002], ['GOTO', None, None, 26], ['VERIFY', 8000, 0, 100], ['+', 8000, 5001, 11000], ['==', '(11000)', 9000, 12001], ['GOTO', 12001, None, 11], ['=', 8000, None, 8002], ['GOTO', None, None, 26], ['VERIFY', 8001, 0, 100], ['+', 8001, 5001, 11001], ['==', '(11001)', 9000, 12002], ['GOTO', 12002, None, 17], ['=', 8001, None, 8002], ['GOTO', None, None, 26], ['ERA', None, None, '~find'], ['+', 8000, 5002, 10003], ['PARAM', 10003, None, 'param0'], ['-', 8001, 5002, 10004], ['PARAM', 10004, None, 'param1'], ['PARAM', 9000, None, 'param2'], ['GOSUB', None, None, '~find'], ['=', 1000, None, 10005], ['=', 10005, None, 8002], ['RETURN', None, None, 8002], ['ENDPROC', None, None, None], ['READ', None, None, 8000], ['=', 5003, None, 8001], ['<', 8001, 8000, 12000], ['GOTO', 12000, None, 38], ['VERIFY', 8001, 0, 100], ['+', 8001, 5001, 11000], ['READ', None, None, '(11000)'], ['+', 8001, 5002, 10001], ['=', 10001, None, 8001], ['GOTO', None, None, 30], ['READ', None, None, 9000], ['ERA', None, None, '~find'], ['PARAM', 5003, None, 'param0'], </pre>

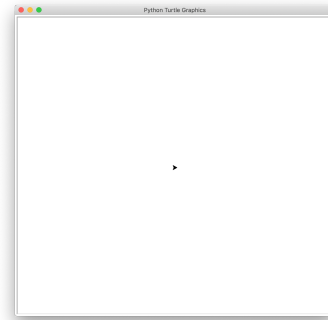
<pre> imprimir &index; } } </pre>	<pre> ['-', 8000, 5002, 10002], ['PARAM', 10002, None, 'param1'], ['PARAM', 9000, None, 'param2'], ['GOSUB', None, None, '~find'], ['=', 1000, None, 10003], ['=', 10003, None, 8002], ['>', 8002, 5000, 12001], ['GOTO', 12001, None, 54], ['PRINT', None, None, 8002], ['VERIFY', 8002, 0, 100], ['+', 8002, 5001, 11000], ['PRINT', None, None, '(11000)'], ['GOTO', None, None, 55], ['PRINT', None, None, 8002] </pre>
Consola	Imagen
<pre> leyendo:5 leyendo:1 leyendo:2 leyendo:3 leyendo:4 leyendo:5 leyendo:6 -1 </pre>	

Sort lineal

Código	Cuádruplos
<pre> programa "sort"; dibujar(){ var &lista : entero[100], &i : entero, &j : entero, &tmp : entero, &n : entero; leer &n; &i=0; </pre>	<pre> ['GOTO', None, None, 1], ['READ', None, None, 8103], ['=', 5000, None, 8100], ['<', 8100, 8103, 12000], ['GOTO', 12000, None, 11], ['VERIFY', 8100, 0, 100], ['+', 8100, 5001, 10001], ['READ', None, None, '(10001)'], ['+', 8100, 5002, 10002], ['=', 10002, None, 8100], ['GOTO', None, None, 3], </pre>

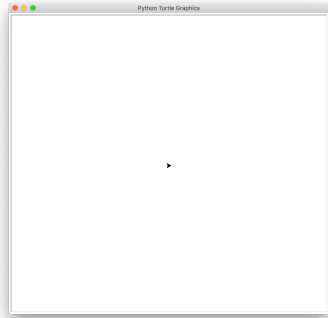
<pre> mientras(&i<&n){ leer &lista[&i]; &i=&i+1; } &i=1; mientras(&i < &n){ &j=0; mientras(&j < &n-1){ si(&lista[&j] > &lista[&j+1]){ &tmp = &lista[&j]; &lista[&j] = &lista[&j+1]; &lista[&j+1] = &tmp; } &j=&j+1; } &i=&i+1; } &i=0; mientras(&i < &n){ imprimir &lista[&i]; &i = &i +1; } } </pre>	<pre> ['=', 5002, None, 8100], ['<', 8100, 8103, 12001], ['GOTO', 12001, None, 44], ['=', 5000, None, 8101], ['-', 8103, 5002, 10006], ['<', 8101, 10006, 12002], ['GOTO', 12002, None, 41], ['VERIFY', 8101, 0, 100], ['+', 8101, 5001, 10007], ['+', 8101, 5002, 10008], ['VERIFY', 10008, 0, 100], ['+', 10008, 5001, 10009], ['>', '(10007)', '(10009)', 12003], ['GOTO', 12003, None, 38], ['VERIFY', 8101, 0, 100], ['+', 8101, 5001, 10010], ['=', '(10010)', None, 8102], ['VERIFY', 8101, 0, 100], ['+', 8101, 5001, 10012], ['+', 8101, 5002, 10013], ['VERIFY', 10013, 0, 100], ['+', 10013, 5001, 10014], ['=', '(10014)', None, '(10012)'], ['+', 8101, 5002, 10016], ['VERIFY', 10016, 0, 100], ['+', 10016, 5001, 10017], ['=', 8102, None, '(10017)'], ['+', 8101, 5002, 10019], ['=', 10019, None, 8101], ['GOTO', None, None, 15], ['+', 8100, 5002, 10021], ['=', 10021, None, 8100], ['GOTO', None, None, 12], ['=', 5000, None, 8100], ['<', 8100, 8103, 12004], ['GOTO', 12004, None, 53], ['VERIFY', 8100, 0, 100], ['+', 8100, 5001, 10024], ['PRINT', None, None, '(10024)'], ['+', 8100, 5002, 10025], ['=', 10025, None, 8100], ['GOTO', None, None, 45] </pre>
<p>Consola</p>	<p>Imagen</p>

leyendo:5
leyendo:1
leyendo:7
leyendo:3
leyendo:4
leyendo:87
1
3
4
7
87



Sort recursivo

Código	Cuádruplos
<pre> programa "find"; var &lista : decimal[100], &tmp : decimal; funcion : void ~bubbleSort(&np : entero){ var &i : entero; si(&np > 1){ &i=0; mientras(&i < &np-1){ si(&lista[&i] > &lista[&i+1]){ &tmp = &lista[&i]; &lista[&i] = &lista[&i+1]; &lista[&i+1] = &tmp; } &i=&i+1; } ~bubbleSort(&np-1); } } dibujar(){ var &n : entero, &i : entero; leer &n; &i=0; mientras(&i<&n){ leer &lista[&i]; </pre>	<pre> ['GOTO', None, None, 35], ['>', 8000, 5000, 12000], ['GOTO', 12000, None, 34], ['=', 5001, None, 8001], ['-', 8000, 5000, 10001], ['<', 8001, 10001, 12001], ['GOTO', 12001, None, 30], ['VERIFY', 8001, 0, 100], ['+', 8001, 5002, 11000], ['+', 8001, 5000, 10002], ['VERIFY', 10002, 0, 100], ['+', 10002, 5002, 11001], ['>', '(11000)', '(11001)', 12002], ['GOTO', 12002, None, 27], ['VERIFY', 8001, 0, 100], ['+', 8001, 5002, 11002], ['=', '(11002)', None, 2100], ['VERIFY', 8001, 0, 100], ['+', 8001, 5002, 11004], ['+', 8001, 5000, 10003], ['VERIFY', 10003, 0, 100], ['+', 10003, 5002, 11005], ['=', '(11005)', None, '(11004)'], ['+', 8001, 5000, 10004], ['VERIFY', 10004, 0, 100], ['+', 10004, 5002, 11007], ['=', 2100, None, '(11007)'], </pre>

<pre> &i=&i+1; } ~bubbleSort(&n); &i=0; mientras(&i<&n){ imprimir &lista[&i]; &i=&i+1; } } </pre>	<pre> ['+', 8001, 5000, 10005], ['=', 10005, None, 8001], ['GOTO', None, None, 4], ['ERA', None, None, '~bubbleSort'], ['-', 8000, 5000, 10007], ['PARAM', 10007, None, 'param0'], ['GOSUB', None, None, '~bubbleSort'], ['ENDPROC', None, None, None], ['READ', None, None, 8000], ['=', 5001, None, 8001], ['<', 8001, 8000, 12000], ['GOTO', 12000, None, 45], ['VERIFY', 8001, 0, 100], ['+', 8001, 5002, 11000], ['READ', None, None, '(11000)'], ['+', 8001, 5000, 10001], ['=', 10001, None, 8001], ['GOTO', None, None, 37], ['ERA', None, None, '~bubbleSort'], ['PARAM', 8000, None, 'param0'], ['GOSUB', None, None, '~bubbleSort'], ['=', 5001, None, 8001], ['<', 8001, 8000, 12001], ['GOTO', 12001, None, 57], ['VERIFY', 8001, 0, 100], ['+', 8001, 5002, 11001], ['PRINT', None, None, '(11001)'], ['+', 8001, 5000, 10003], ['=', 10003, None, 8001], ['GOTO', None, None, 49] </pre>
Consola	Imagen
<pre> leyendo:5 leyendo:29 leyendo:4 leyendo:69 leyendo:4 leyendo:8 4.0 4.0 8.0 29.0 69.0 </pre>	

Dibujar objetos

Código	Cuádruplos
<pre> programa "prueba"; var &i : entero, &j : decimal, &x : entero, &y : entero, &arreglo : entero[10], &berny : entero; funcion : entero ~factorial(&ii : entero){ var &tmp : entero; si(&ii >= 1){ &tmp = &ii*~factorial(&ii-1); }sino{ &tmp = 1; } regresar &tmp; } funcion : void ~print(&t : entero){ si(&t>=1){ imprimir &t; ~print(&t-1); imprimir &t; } } funcion : entero ~uno(&juan : entero, &pancho : entero){ var &k : decimal; &k = 1 + 2 - -3 * (-3.0 - 4 + 5); si(&juan < &pancho){ imprimir &i; } sino { imprimir &j; } regresar &i; } </pre>	<pre> ['GOTO', None, None, 52], ['>=', 8000, 5000, 12000], ['GOTO', 12000, None, 11], ['ERA', None, None, '~factorial'], ['-', 8000, 5000, 10000], ['PARAM', 10000, None, 'param0'], ['GOSUB', None, None, '~factorial'], ['=', 1014, None, 10001], ['*', 8000, 10001, 10002], ['=', 10002, None, 8001], ['GOTO', None, None, 12], ['=', 5000, None, 8001], ['RETURN', None, None, 8001], ['ENDPROC', None, None, None], ['>=', 8000, 5000, 12000], ['GOTO', 12000, None, 22], ['PRINT', None, None, 8000], ['ERA', None, None, '~print'], ['-', 8000, 5000, 10000], ['PARAM', 10000, None, 'param0'], ['GOSUB', None, None, '~print'], ['PRINT', None, None, 8000], ['ENDPROC', None, None, None], ['+', 5000, 5001, 10000], ['-', 6000, 5002, 11000], ['+', 11000, 5003, 11001], ['*', 5004, 11001, 11002], ['-', 10000, 11002, 11003], ['=', 11003, None, 9000], ['<', 8000, 8001, 12000], ['GOTO', 12000, None, 33], ['PRINT', None, None, 1000], ['GOTO', None, None, 34], ['PRINT', None, None, 2000], ['RETURN', None, None, 1000], ['ENDPROC', None, None, None], ['=', 5005, None, 8001], ['<', 8000, 9000, 12000], ['GOTO', 12000, None, 41], ['PRINT', None, None, 1000], ['GOTO', None, None, 42], ['PRINT', None, None, 2000], ['ENDPROC', None, None, None], ['ERA', None, None, '~uno'], ['PARAM', 8000, None, 'param0'], ['PARAM', 8001, None, 'param1'], ['GOSUB', None, None, '~uno'], ['=', 1015, None, 10000], ['RETURN', None, None, 10000], </pre>

```

funcion : void ~dos(&juan : entero,
&pancho : decimal){
    var &k : entero;

    &k = 1283;

    // imprimir &k;
    // imprimir (1+2-3*(4/3*4.3));

    si(&juan < &pancho){
        imprimir &i;
    } sino {
        imprimir &j;
    }
}

funcion : entero ~tres(&juan : entero,
&pancho : entero){
    regresar ~uno(&juan, &pancho);
}

funcion : entero ~cero(){
    regresar 0;
}

dibujar(){
    var &h : decimal, &turtle : entero;

    &i = 120;
    &j = 50.34;
    &x = 4;
    &y = 7;
    &h = 123.23;
    &turtle = 1;

    leer &j;
    imprimir &j;
    leer &turtle;
    imprimir &turtle;

    adelante 60;
    rotar &i;
    adelante 60;
    cuadrado 100, 60, 19, rgb(10, 34,
255), 5;
    circulo -100, -60, 10, rgb(10,34,23),

```

```

['ENDPROC', None, None, None],
['RETURN', None, None, 5006],
['ENDPROC', None, None, None],
['=', 5007, None, 1000],
['=', 6001, None, 2000],
['=', 5002, None, 1001],
['=', 5008, None, 1002],
['=', 6002, None, 9000],
['=', 5000, None, 8000],
['READ', None, None, 2000],
['PRINT', None, None, 2000],
['READ', None, None, 8000],
['PRINT', None, None, 8000],
['MOVF', None, None, 5009],
['ROT', None, None, 1000],
['MOVF', None, None, 5009],
['SCUAD', None, None, None],
['PARAM', 5010, None, 'param0'],
['PARAM', 5009, None, 'param1'],
['PARAM', 5011, None, 'param2'],
['PARAM', 7000, None, 'param3'],
['PARAM', 5003, None, 'param4'],
['ECUAD', None, None, None],
['SCIR', None, None, None],
['PARAM', 5012, None, 'param0'],
['PARAM', 5013, None, 'param1'],
['PARAM', 5014, None, 'param2'],
['PARAM', 7001, None, 'param3'],
['PARAM', 5001, None, 'param4'],
['ECIR', None, None, None],
['STRI', None, None, None],
['PARAM', 2000, None, 'param0'],
['PARAM', 5014, None, 'param1'],
['ERA', None, None, '~uno'],
['PARAM', 5009, None, 'param0'],
['ERA', None, None, '~tres'],
['PARAM', 5000, None, 'param0'],
['PARAM', 5004, None, 'param1'],
['GOSUB', None, None, '~tres'],
['=', 1016, None, 10000],
['PARAM', 10000, None, 'param1'],
['GOSUB', None, None, '~uno'],
['=', 1015, None, 10001],
['PARAM', 10001, None, 'param2'],
['PARAM', 5003, None, 'param3'],
['PARAM', 7002, None, 'param4'],
['PARAM', 5001, None, 'param5'],
['ETRI', None, None, None],
['STXT', None, None, None],
['PARAM', 5012, None, 'param0'],
['PARAM', 5009, None, 'param1'],
['PARAM', 7003, None, 'param2'],
['PARAM', 7004, None, 'param3'],

```

```

2;
  triangulo &j, 10, ~uno(60,
~tres(1,-3)), 5, rgb(10, 34, 255), 2;
  texto -100, 60, 'juancho',
rgb(255,34,23), 24;
  adelante 60;
  rotar &i;
  adelante 60;

&i = 1;

mientras(&i < 10){
  &i = &i + 1;
  imprimir &i;
  ~dos(&x, 1+2-3/4*4*(4+3/3*&x));
  imprimir &i;
}

&arreglo[1] = 9999;
imprimir &arreglo[1];

&berny = &arreglo[1];
imprimir &berny;

&arreglo[&arreglo[1]-9997] = 2;
imprimir &arreglo[2];

&arreglo[~cero()] = -1;
imprimir &arreglo[0];

mientras(&turtle < 100){
  adelante 90;
  rotar 89;
  adelante 1+&turtle;
  &turtle = &turtle+1;
}

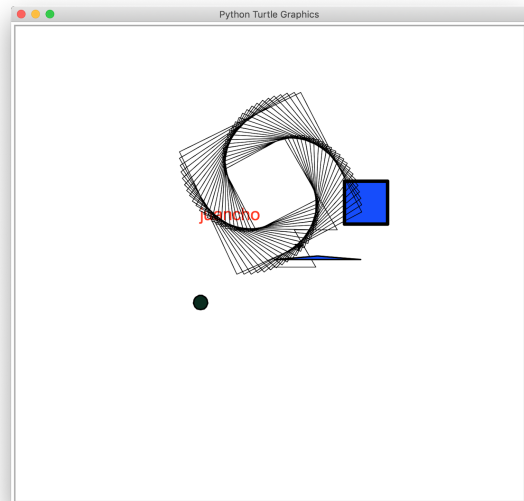
&i = ~uno(&x, &y);
imprimir &i;
imprimir ~factorial(6);
~print(3);
}

```

```

['PARAM', 5015, None, 'param4'],
['ETXT', None, None, None],
['MOVF', None, None, 5009],
['ROT', None, None, 1000],
['MOVF', None, None, 5009],
['=', 5000, None, 1000],
['<', 1000, 5014, 12000],
['GOTO', 12000, None, 127],
['+', 1000, 5000, 10003],
['=', 10003, None, 1000],
['PRINT', None, None, 1000],
['ERA', None, None, '~dos'],
['PARAM', 1001, None, 'param0'],
['+', 5000, 5001, 10005],
['/', 5016, 5002, 11000],
['*', 11000, 5002, 11001],
['/', 5016, 5016, 11002],
['*', 11002, 1001, 11003],
['+', 5002, 11003, 11004],
['*', 11001, 11004, 11005],
['-', 10005, 11005, 11006],
['PARAM', 11006, None, 'param1'],
['GOSUB', None, None, '~dos'],
['PRINT', None, None, 1000],
['GOTO', None, None, 108],
['VERIFY', 5000, 0, 10],
['+', 5000, 5017, 10006],
['=', 5018, None, '(10006)'],
['VERIFY', 5000, 0, 10],
['+', 5000, 5017, 10008],
['PRINT', None, None, '(10008)'],
['VERIFY', 5000, 0, 10],
['+', 5000, 5017, 10009],
['=', '(10009)', None, 1013],
['PRINT', None, None, 1013],
['VERIFY', 5000, 0, 10],
['+', 5000, 5017, 10011],
['-', '(10011)', 5019, 10012],
['VERIFY', 10012, 0, 10],
['+', 10012, 5017, 10013],
['=', 5001, None, '(10013)'],
['VERIFY', 5001, 0, 10],
['+', 5001, 5017, 10015],
['PRINT', None, None, '(10015)'],
['ERA', None, None, '~cero'],
['GOSUB', None, None, '~cero'],
['=', 1017, None, 10016],
['VERIFY', 10016, 0, 10],
['+', 10016, 5017, 10017],
['=', 5020, None, '(10017)'],
['VERIFY', 5006, 0, 10],
['+', 5006, 5017, 10019],
['PRINT', None, None, '(10019)'],

```


	<pre> ['<', 8000, 5010, 12001], ['GOTO', 12001, None, 164], ['MOV', None, None, 5021], ['ROT', None, None, 5022], ['+', 5000, 8000, 10020], ['MOV', None, None, 10020], ['+', 8000, 5000, 10021], ['=', 10021, None, 8000], ['GOTO', None, None, 155], ['ERA', None, None, '~uno'], ['PARAM', 1001, None, 'param0'], ['PARAM', 1002, None, 'param1'], ['GOSUB', None, None, '~uno'], ['=', 1015, None, 10023], ['=', 10023, None, 1000], ['PRINT', None, None, 1000], ['ERA', None, None, '~factorial'], ['PARAM', 5023, None, 'param0'], ['GOSUB', None, None, '~factorial'], ['=', 1014, None, 10025], ['PRINT', None, None, 10025], ['ERA', None, None, '~print'], ['PARAM', 5016, None, 'param0'], ['GOSUB', None, None, '~print'] </pre>
Consola	Imagen
<pre> leyendo:3 3.0 leyendo:2 2 3.0 120 2 3.0 2 3 3.0 3 4 3.0 4 5 3.0 5 6 3.0 </pre>	

6	
7	
3.0	
7	
8	
3.0	
8	
9	
3.0	
9	
10	
3.0	
10	
9999	
9999	
2	
-1	
10	
10	
720	
3	
2	
1	
1	
2	
3	

Listados documentados del proyecto

main.py

```
tree_parsed = daxe_parser.parse(parse_str.read())
```

Es el trigger del parser, toma como parámetros, string en este caso el archivo como string, ejecuta el visitor.py por defecto y se recorre cada nodo que esté declarado en el visitante para poder hacer ejecución de la función

Visitor.py

Visita todos los puntos neurálgicos declarados en el parser, recorre el árbol de parseo de manera bottom up, si encuentra un punto neurálgico declarado como función la ejecuta. todas las funciones toman como parámetros los hijos de ese nodo, para poder obtener información específica de los tokens terminales.

```
class DaxeVisitor(Visitor Recursive):
```

quadruplets.py

algorithm_with

toma como parámetros un arreglo de operadores ejemplo ["="] o ["/", "*"] entra si el tope del operador está en la lista, después se comparan los tipos para ver que no generen error generan la variable temporal con la dirección correcta

función principal de cuádruplos para poder generar expresiones (suma, resta, división, relacional, asignación) se accede desde visitor, en los puntos neurálgicos.

```
def algorithm_with(self, labels):
    """
    Algorithm_with: toma como parametros un arreglo de operadores
    ejemplo ["="] o ["/", "*"]

    entra si el tope del operador esta en la lista,
    despues se comparan los tipos para ver que no generen error
    generan la variable temporal con la direccion correcta
    """
    operator = self.operators.top()
    if operator in labels:
        right_operand = self.operands.pop()
        right_type = self.types.pop()
        left_operand = self.operands.pop()
        left_type = self.types.pop()
        operator = self.operators.pop()
        result_type = self.semantic_cube.cube[left_type][right_type][operator]
        if result_type == "ERROR":
            print(left_operand, left_type, right_operand, right_type)
            raise Exception("Tipos no coinciden en la %s (tipos: %s) a %s (tipos: %s), en:
%s:%s"%(operator, right_type, left_operand, left_type, left_operand.line,
left_operand.column))
        else:
            self.num_aviables+=1
            result_name = Token("T_TMP_ID", 'tmp_'+str(result_type)+'_'+str(self.num_aviables),
line=left_operand.line, column= left_operand.column)
            # print(result_name.value)
            left = self.token_to_dir(left_operand)
            right = self.token_to_dir(right_operand)
            result = self.token_to_dir(result_name)
            self.gen_quad(operator, left, right, result)
            # self.gen_quad(operator, left_operand.value, right_operand.value, result_name.value)
            if operator != "=":
                self.operands.push(result_name)
                self.types.push(result_type)
```

token_to_dir

token una variable tipo token tiene propiedad de tipo y valor, look_in es un directorio de variables si se proporciona busca ahi si no lo busca en la función actual

switch de tipo de token para saber que contador aumentar y que tipo de variable es si es flotante, booleano, entero, constante, string.

se accede internamente en la clase de cuádruplos para poder generar nombres de temporales a dirección, es el convertidos de ids a números

```
def token_to_dir(self, token, look_in = None):
    """
    token_to_dir: toke una variable tipo token
    tiene propiedad de tipo y valor, look_in es un directorio de variables
    si se porporciona busca ahi si no lo busca en la funcion actual

    switch de tipo de token para saber que contador aumentar y que tipo
    de variable es si es flotante, booleano, entero, contante, string
    """
    value = None
    if token.type == "T_NUM_INT":
        tmp = self.memory.search(int(token.value), 5)
        if tmp == None:
            value = self.cte_int
            self.cte_int+=1
            self.memory.add(int(token.value), value)
        else:
            value=tmp

    elif token.type == "T_NUM_FLOAT":
        tmp = self.memory.search(float(token.value), 6)
        if tmp == None:
            value = self.cte_decimal
            self.cte_decimal+=1
            self.memory.add(float(token.value), value)
        else:
            value = tmp

    elif token.type == "T_TMP_ID":
        if token.value in self.dir_tmp:
            value = self.dir_tmp[token.value]
        else:
            if "entero" in token.value:
                self.dir_tmp[token.value] = self.tmp_int
                value = self.tmp_int
                self.tmp_int+=1

            if "decimal" in token.value:
                self.dir_tmp[token.value] = self.tmp_decimal
                value = self.tmp_decimal
                self.tmp_decimal+=1

            if "booleano" in token.value:
                self.dir_tmp[token.value] = self.tmp_bool
```

```

        value = self.tmp_bool
        self.tmp_bool+=1

    elif token.type == "T_VAR_ID":
        if look_in != None:
            value = look_in[token.value]['dirV']
        else:
            value = self.fun_dir.get_dirV_of(token.value)

    elif token.type == 'T_FUN_ID':
        fun_table = self.fun_dir.get_fun_table_by_id(token.value)
        if fun_table["type"] == 'entero':
            value = self.glo_tmp_int
            self.glo_tmp_int+=1

        elif fun_table["type"] == 'decimal':
            value = self.glo_tmp_decimal
            self.glo_tmp_decimal+=1

    elif token.type == 'T_ID':
        value = self.cte_string
        self.cte_string+=1
        self.memory.add(token.value, value)

    elif token.type == 'T_COLOR':
        value = self.cte_string
        self.cte_string+=1
        self.memory.add(token.value, value)

    elif token.type == 'T_TMP_DIR':
        value = token.value

    return value

```

gen_array_access

Sacar los id del array y su expresión interna para acceso si el index no es entero marca error generan la suma de la dirección base más el tamaño del temporal para tener la dirección y generar su temporal (T) y agregarlo a los operadores

se utiliza desde la clase de visitante para poder recorrerlo y cuando toque el punto neurálgico del right paréntesis del arreglo, se genera esta llamada.

```

def generate_array_access(self):
    """
        Sacar los id del array y su expresion interna para acceso

        si el index no es entero marca error
    """

```

```

        generan la suma de la direccion base mas el tamano del temporal
        para tener la direccion y generar su temporal (T)
        y agregarlo a los operadores
    """
    aux = self.operands.pop()
    aux_type = self.types.pop()
    array_id = self.operands.pop()
    array_type = self.types.pop()

    if aux_type != "entero":
        raise Exception("El \xc3\xadndice del arreglo (%s) no es un entero en
%s:%s"%(aux.value, aux.line, aux.column))

    base = self.curren_arr["dirV"]

    self.num_aviables+=1
    result_name = Token("T_TMP_ID", 'tmp_'+str(array_type)+'_'+str(self.num_aviables),
line=aux.line, column= aux.column)
    base_name = Token("T_NUM_INT", str(base), line=aux.line, column= aux.column)

    left = self.token_to_dir(aux)
    result = self.token_to_dir(result_name)
    dir_base = self.token_to_dir(base_name)
    self.gen_quad("+", left, dir_base, result)

    tmp_token = Token("T_TMP_DIR", '('+str(result)+')', line=aux.line, column= aux.column)
    self.operands.push(tmp_token)
    self.types.push(array_type)
    self.operators.pop()

```

function_dir.py

create_function_vars

Agrega la función actual declarada, con sus variables y parámetros vacíos, en caso de que no sea tipo void, genera una propiedad return para, generar una dirección virtual

se manda llamar desde el visitor en el punto neurálgico cuando se tiene el nombre y el tipo de la función.

```

def create_function_vars(self):
    """
    Agrega la funcion actual declarada
    con sus varibales y parametros vacios,
    en caso de que no sea tipo void
    genera una propiedad return para
    generar una direccion virtual
    """
    self.dir[self.current_func]={

```

```

        "type": self.current_type,
        "vars": OrderedDict(),
        "params": OrderedDict()
    }
    if self.current_type != "void":
        self.dir[self.current_func]["return"]=self.get_aviable_dir(self.current_type, True)

```

add_variable

busca en las variables locales y globales si están declaradas si ya lo están regresa un error y crea una propiedad de variable actual y guarda su tipo y dirección virtual dependiendo los contadores en donde estén.

se manda llamar desde el visitor cuando detecta el tipo y el nombre de la variables y se agrega a la función actualmente activa.

```

def add_variable(self, var_name):
    """
        busca en las variables locales y globales si
        estan declaradas si ya lo estan regresa un error
        y crea una propiedad de varaible actual y guarda
        su tipo y direccion virtual dependiendo los contadores
        en donde esten.
        var_name: "string"
    """
    if var_name in self.dir[self.current_func]["vars"] or ("vars" in
self.dir[self.program_key] and var_name in self.dir[self.program_key]["vars"]):
        raise Exception("Declaraci\u00f3n m\u00faltiple de variable(%s) en la
funci\u00f3n(%s)"%(var_name, self.current_func))
    self.current_var = var_name
    self.dir[self.current_func]["vars"][var_name]={
        "type": self.current_type,
        "dirV": self.get_aviable_dir(self.current_type, self.current_func == self.program_key)
    }

```

add_function

detecta si la función ya ha sido declarada, en ese caso regresa error en caso contrario crea un directorio para las funciones sin propiedades.

Se accede desde el visitor para poder agregar una función cuando se llega el punto neurálgico adecuado.

```

def add_function(self, name):
    """
        detecta si la funcion ya ha sido declarada, en ese caso
        regresa error en caso contrario crea un directorio
        para las funciones sin propiedades.
    """

```

```

        name: "nombre de la funcion"
        """
        if name in self.dir:
            raise Exception("Declaraci\u00f3n m\u00faltiple de funciones con el mismo nombre
%s"%(name))
        self.dir[name]={}
        self.current_func=name

```

custom_stack.py

pop_arg

se hace un loop para poder agregar en el arreglo temporal y se saca del arreglo de instancia para borrar de la memoria de parámetros, y regresar los que se ocupa.

se ingresa desde la clase de cuádruplos para poder generar la recursión de funciones.

```

def pop_arg(self):
    """
    loop por el stack hasta que detecte el fondo falso
    cada loop generar un pop (desechar informacion)
    """
    tmp = Stack()
    for i in reversed(self.stack):
        if i != "(":
            tmp.push(self.pop())
        else:
            self.pop();
            break
    return tmp;

```

top_arg

loop en el stack (en reversa para ir del tope para abajo) después se lo agregan a un stack temporal y regresa toda esa info.

```

def top_arg(self):
    """
    loop en el stack para generar un stack temporal
    sin sacarlos en el general
    """
    tmp = Stack()
    for i in reversed(self.stack):
        if i != "(":
            tmp.push(i)
        else:
            break
    return tmp;

```


exe_memory.py

get

checas si es un string para ver si es una dirección de arreglo en caso de serlo obtiene el valor de la dirección y substituye la direccion_virtual obtenida, convierte la dirección virtual en manera accesible para la memoria y regresa su valor, en caso de que la dirección no existe arroja variable no inicializada

se accede desde la máquina virtual para obtener valores en manera de ejecución

```
def get(self, dir_v):
    """
    checas si es un string para ver si es una direccion de arreglo
    en caso de serlo obtiene el valor de sa direccion
    y substituye la direccion_virtual obtenida,

    combierte la direccion virtual en manera accesibe para la memoria
    y regresa su valor, en caso de que la direccio no existe arroja
    variable no inicializada
    """
    try:
        if isinstance(dir_v, str):
            dir_str = re.findall('\d+|$', dir_v)[0]
            dir_int = int(dir_str)
            dir_v = self.get(dir_int)
        key1 = dir_v//1000
        key2 = dir_v%1000
        # pprint.pprint(self.directions)
        # print(self.directions[key1][key2])
        return self.directions[key1][key2]
    except KeyError:
        raise Exception("Variable no inicializada")
```

add

en caso de ser un string convierte el número a dirección para poder traer la direccion donde esta guardado el valor, después convierte la dirección virtual a número accesible en memoria y le asigna el valor pasado como parámetro

se accede desde la máquina virtual para poder agregar valores en tiempo de ejecución.

```
def add(self, value, dir_v):
    """
    en caso de ser un string convierte el numero a direccion para poder
    traer la direccion donde esta guardado el valor,

    despues convierte la direccion virtual a numero accesible en memoria
    y le asigna el valor pasado como parametro
    value: el valor a asignar 4.3
    dir_v: direccion en donde asignarla (1000)
```

```
"""
if isinstance(dir_v, str):
    dir_str = re.findall('\d+|$', dir_v)[0]
    dir_int = int(dir_str)
    dir_v = self.get(dir_int)
key1 = dir_v/1000
key2 = dir_v%1000
self.directions[key1][key2] = value
```