> **Universidad de San Carlos de Guatemala**
>
> **Facultad de ingeniería**
>
> **Ingeniería en Ciencias y Sistemas**

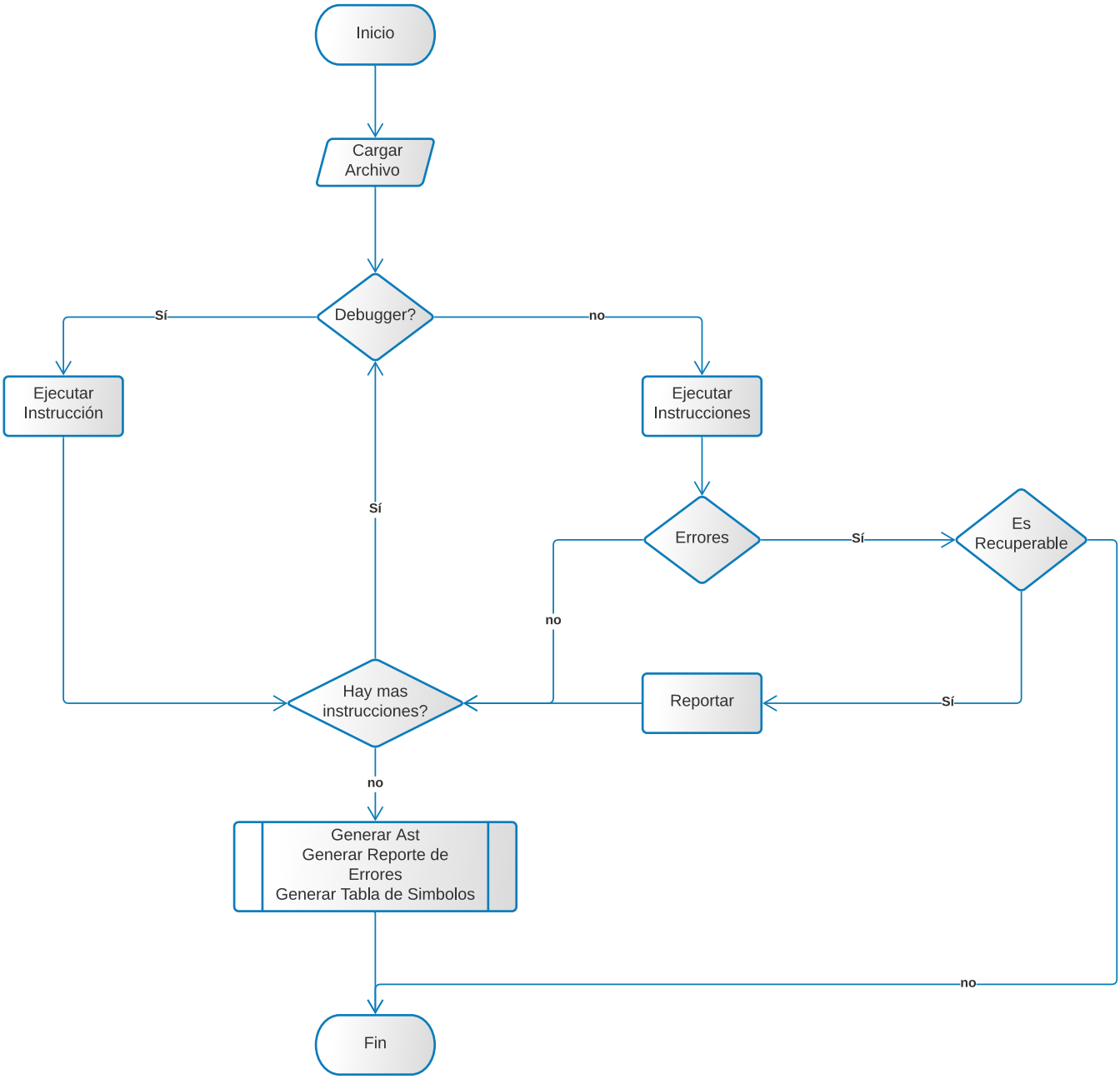
Manual de Técnico

> **Elian Saúl Estrada Urbina**
>
> **201806838**

Especificaciones Técnicas para correr la aplicación

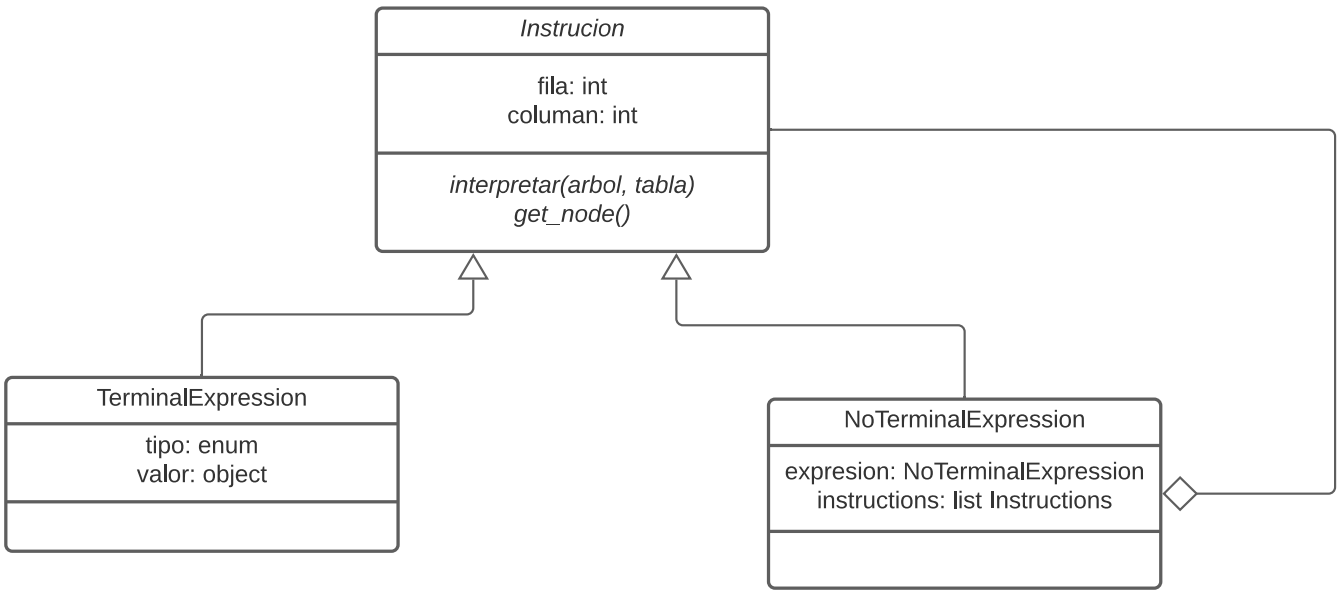
Sistema Operativo	Arquitectura	Versión	Memoria RAM
Windows	64 bits	Windows7 SP3 o superior	mínimo 2GB de memoria RAM
Linux	64 bits	Distribucines Debian 9 o superior	mínimo 2GB de memoria RAM

Flujo



Patrón

El patrón utilizado para la implementación del proyecto fue el patrón Interprete:



Explicación Clases

Nombre	Parametros	Métodos	Explicación
Instruction	row y column	interpretar(tree, table) get_node()	Clase abstracta padre de todas las clases que utilizaremos como instrucción
AST_Node	value	add_child(string) add_childs_node(list childs)	Clase para crear el arbol para graficar el ast.
type	Enum	--	Clase enumerada para los tipos de datos
Arithmetic_Operator	Enum	--	Clase enumerada para los operadores aritméticos
Relational_Operator	Enum	--	Clase enumerada para los operadores relacionales
Logical_Operator	Enum	--	Clase enumerada para los operadores lógicos

Nombre	Parametros	Métodos	Explicación
Tree	instructions	get_dot(root) travel_ast(id_root, node_root)	Clase que alberga todas las instrucciones generadas por el analizador, también genera el grafico del ast y al macena información para la tabla de simbolos y el debugger
SymbolTable	prev = None, name = "Globa"	set_table(symbol) get_table(id) update_table(symbol)	Clase que alberta todas las tablas de simbolos, funciona en su mayoría de veces como una lista de tablas, pudiendo acceder a la tabla anterior, esto con el fin de manejar los ambitos.
Symbol	id, type, row, column, value, environment	setters and getters	Clase para almacenar todas las variables, arreglos etc...

Nombre	Parametros	Métodos	Explicación
Error	type, description, row, column	setters an d getters str()	Clase que maneja los errores de tipo lexico, sintacitoc, semantico
Primitive	type, value, row, column	interpret(tree, table) get_node()	Clase que interpreta los datos primitivos es la expresión mas pequeña.
Identifier	id, row, column	interpret(tree, table) get_node()	Clase que interpreta los identificadores y devuelve un valor.
Arithmetic	exp1, exp2, operator, row, column	interpret(tree, table) get_node() get_operator(operator)	Clase que se encarga de interpretar las expresiones aritmeticas, vease sumas, restas, multiplicaciones etc.

Nombre	Parametros	Métodos	Explicación
Relational	exp1, exp2, operator, row, column	interpret(tree, table) get_node() get_operator(operator)	Clase que se encarga de interpretar las expresiones relacionales y devolver un valor booleano, vease como dichas operaciones, mayor, igual, diferente, etc.
Logical	exp1, exp2, operator, row, column	interpret(tree, table) get_node() get_operator(operator)	Clase que se encarga de interpretar las expresiones logicas como and, or y not y devolver un valor booleano.
Casting	type, exp, row, column	interpret(tree, table) get_node()	Clase que se encarga de hacer conversiones de tipos de datos.
Read	row, column	interpret(tree, table) get_node()	Clase que se encarga de leer datos del usuario para nuestra aplicación

Nombre	Parametros	Métodos	Explicación
Array	type_init, len_init, name, type_assig, expression, list_expression, row, column, id_array = None	interpret(tree, table) get_node() get_values(list_values, tree, table) get_graph_expression(node, list_value) get_dimensions(node) get_list(list_values, tree, table)	Clase que se encarga de declarar arreglos de los 3 tipos utilizados en esta aplicación.
Access_Array	name, position, expression, row, column	interpret(tree, table) get_node() lexico_map(positions, max_index, tree, table) value_positions(positions, list_values, tree, table) assign_value(positions, list_values, flag = False, flag2 = True)	Clase encargada de manejar los accesos a arreglos, ya se para devolver un valor en la posición especificada o para asignar un nuevo valor en una posición específica.
Assignment	id, expression, row, column	interpret(tree, table) get_node()	Clase encargada de manejar las asignaciones a variables previamente declaradas.
Break	row, column	interpret(tree, table) get_node()	Clase que interrumpe un ciclo o las instrucciones de un switch

Nombre	Parametros	Métodos	Explicación
Call	name, params, row, column	interpret(tree, table) get_node()	Clase que maneja las llamadas de funciones, pasando los parametros y ejecutando las instrucciones de dicha función.
Case	exp, instructions, row, column	interpret(tree, table) get_node()	Clase que alberga las instrucciones de un case de switch, para posteriormente ser ejecutadas si su expresión hace match con la del switch
Continue	row, column	interpret(tree, table) get_node()	Clase que se ignora las instrucciones posteriores en un ciclo y avanza a la siguiente iteración.
Declaration	id, row, column, expression = None	interpret(tree, table) get_node()	Clase que se encarga de creer una nueva vairable y almacenarla en la tabla de simbolos para su posterior uso.

Nombre	Parametros	Métodos	Explicación
For	init, condition, advance, instructions, row, column	interpret(tree, table) get_node()	Clase que ejecuta un ciclo for dependiendo si su condición es verdadera, tiene un bloque de inicialización y uno de avance.
Function	name, params, instructions, row, column	interpret(tree, table) get_node()	Clase que se encarga de declarar una función y almacenarla en el arbol para su posterior uso.
If	exp, instructions, else_instructions, else_if, row, column	interpret(tree, table) get_node()	Esta clase interpreta las instrucciones del if si su primra condicion es verdadera, de lo contrario buscara una instrucción else o else-if y si las ecuentra procedera a ejecutar sus instrucciones.

Nombre	Parametros	Métodos	Explicación
Inc_Dec	exp, operator, row, column	interpret(tree, table) get_node()	Esta clase se encarga de aumentar o disminuir el valor de una variable en 1 dependiendo del operador que se le asigne.
Main	instructions, row, column	interpret(tree, table) get_node()	Clase principal de nuestro proyecto es la que albergara todas las instrucciones ejecutables de nuestro programa.
Print	expression, row, column	interpret(tree, table) get_node()	Imprime la expresión en nuestra consola de salida.
Return	expression, row, column	interpret(tree, table) get_node()	Retorna el valor de la expresión y sirve para interrumpir el flujo de una función.

Nombre	Parametros	Métodos	Explicación
Switch	exp, list_case, default, row, column	interpret(tree, table) get_node() execute_instructions	Clase que valida si su expresión es igual a alguno de sus casos, si una hace match ejecuta sus instrucciones, si en dado caso ninguna hace match y se tiene un default ejecuta el default si no solo saldra.
While	exp, instructions, row, column	interpret(tree, table) get_node()	Clase que ejecutara sus instrucciones siempre que su condición sea verdadera una y otra vez, si es falsa no las ejecutara nunca.