Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Organización de Lenguajes y Compiladores 2

Catedrático: M.Sc. Luis Espino

Auxiliares: Juan Carlos Maeda, Pavel Vásquez

**AUGUS**

**Manual Técnico**

Kairi Itzmar Alexander Franco Medina

201222591

17 de junio de 2020

ÍNDICE

[INTRODUCCIÓN 2](#_Toc43289036)

[Objetivo general 2](#_Toc43289037)

[Objetivo específico 2](#_Toc43289038)

[RESPONSABLES 2](#_Toc43289039)

[CONTENIDO TÉCNICO 3](#_Toc43289040)

[Clases 3](#_Toc43289041)

[Gramática 3](#_Toc43289042)

# INTRODUCCIÓN

El lenguaje de programación *Augus* está basado e PHP y en MIPS, cuya finalidad principal es ser un lenguaje intermedio. Augus tiene dos restricciones, cada instrucción es una operación, con un máximo de dos operandos en cada una. Augus reconoce cuatro tipos de datos: entero, punto flotante, cadena de caracteres y arreglo; no utiliza ciclos for, while, ni do-while.

Dentro de la aplicación se pueden realizar diferentes acciones como:

El siguiente manual le será útil principalmente a los programadores para poder darle mantenimiento a la aplicación o revisarla en caso de alguna falla.

## Objetivo general

Interpretar código abierto para un lenguaje.

## Objetivo específico

Interpretar el lenguaje Augus mediante la traducción dirigida por la sintaxis, utilizando PLY Python.

# RESPONSABLES

Asignación del proyecto: M.Sc. Luis Espino

Programador: Kairi Itzmar Alexander Franco Medina

Encargados de revisión: Juan Carlos Maeda, Pavel Velasquez

# CONTENIDO TÉCNICO

## Clases

Grammar:

Clase que contiene la definición de los tokens y de la gramática mediante PLY, y la función recursiva para el recorrido del árbol. La gramática se encuentra en el área de anexos.

Analyze(String): recibe la cadena de entrada y realiza el análisis léxico y sintáctico

run(tree): este método recibe tuplas del AST y las recorre, ejecutando el análisis semántico y realizando las acciones correspondientes.

Symbol:

Clase que maneja los símbolos que luego se guardarán en la tabla de símbolos. Tiene los atributos “id”, “value”, “varType”, “tree”. La variable “tree” es el árbol asociado al símbolo, en caso sea un salto condicional o incondicional. Los métodos son:

Table:

Es una lista de elementos de la clase Symbol. Representa la tabla de símbolos.

isSymbolInTable(id): verifica que el id exista dentro de la tabla de símbolos. Retorna True si existe.

Add(Symbol): Agrega un símbolo a la tabla. Si ya existe, reemplaza el valor.

Get(id): retorna el símbolo que corresponda con el id.

Remove(id): elimina el registro de la tabla de símbolos que corresponda al id dado.

Error:

Clase que maneja los diferentes tipos de erroes. Cada error es un elemento de una lista de errores. Tiene los atributos “error”, “line”, “column”.

ErrorList:

Lista de objetos de tipo error.

Add(Error): agrega un error a la lista.

Clear(): elimina todos los errores de la lista

## Gramática

El análisis léxico y sintáctico fue realizado utilizando la librería de Python PLY. La gramática utilizada para el análisis es la siguiente.

start -> MAIN COLON body

body -> goto body

| unset body

| print body

| exit body

| tag body

| if body

| assign body

| declaration body

body -> empty

goto -> GOTO NAME SEMICOLON

unset -> UNSET L\_PAR TVAR R\_PAR SEMICOLON

| UNSET L\_PAR AVAR R\_PAR SEMICOLON

| UNSET L\_PAR VVAR R\_PAR SEMICOLON

| UNSET L\_PAR SVAR R\_PAR SEMICOLON

| UNSET L\_PAR SPVAR R\_PAR SEMICOLON

| UNSET L\_PAR RAVAR R\_PAR SEMICOLON

print -> PRINT L\_PAR TVAR R\_PAR SEMICOLON

| PRINT L\_PAR AVAR R\_PAR SEMICOLON

| PRINT L\_PAR VVAR R\_PAR SEMICOLON

| PRINT L\_PAR SVAR R\_PAR SEMICOLON

| PRINT L\_PAR SPVAR R\_PAR SEMICOLON

| PRINT L\_PAR RAVAR R\_PAR SEMICOLON

print -> PRINT L\_PAR TVAR L\_BRACKET arithmetic R\_BRACKET R\_PAR SEMICOLON

| PRINT L\_PAR AVAR L\_BRACKET arithmetic R\_BRACKET R\_PAR SEMICOLON

| PRINT L\_PAR VVAR L\_BRACKET arithmetic R\_BRACKET R\_PAR SEMICOLON

| PRINT L\_PAR SVAR L\_BRACKET arithmetic R\_BRACKET R\_PAR SEMICOLON

| PRINT L\_PAR SPVAR L\_BRACKET arithmetic R\_BRACKET R\_PAR SEMICOLON

| PRINT L\_PAR RAVAR L\_BRACKET arithmetic R\_BRACKET R\_PAR SEMICOLON

print -> PRINT L\_PAR STRING R\_PAR SEMICOLON

exit -> EXIT SEMICOLON

tag -> NAME COLON

if -> IF L\_PAR condition R\_PAR GOTO NAME SEMICOLON

condition -> condition AND condition

| condition OR condition

condition -> NOT condition

condition -> relational

relational -> arithmetic EQUAL arithmetic

| arithmetic NOT\_EQUAL arithmetic

| arithmetic GREATER arithmetic

| arithmetic LESS arithmetic

| arithmetic GREATER\_EQUAL arithmetic

| arithmetic LESS\_EQUAL arithmetic

relational -> arithmetic

arithmetic -> arithmetic PLUS arithmetic

| arithmetic MINUS arithmetic

| arithmetic MULTIPLY arithmetic

| arithmetic DIVIDE arithmetic

| arithmetic REMAINDER arithmetic

arithmetic -> INTEGER

| DECIMAL

| STRING

| var

| array\_access

arithmetic -> MINUS arithmetic

var -> TVAR

| AVAR

| VVAR

| SVAR

| RAVAR

| SPVAR

array\_access ->

assign -> TVAR ASSIGN condition SEMICOLON

| AVAR ASSIGN condition SEMICOLON

| VVAR ASSIGN condition SEMICOLON

| SVAR ASSIGN condition SEMICOLON

assign -> TVAR ASSIGN conversion SEMICOLON

| AVAR ASSIGN conversion SEMICOLON

| VVAR ASSIGN conversion SEMICOLON

| SVAR ASSIGN conversion SEMICOLON

assign -> TVAR ASSIGN READ L\_PAR R\_PAR SEMICOLON

| AVAR ASSIGN READ L\_PAR R\_PAR SEMICOLON

| VVAR ASSIGN READ L\_PAR R\_PAR SEMICOLON

| SVAR ASSIGN READ L\_PAR R\_PAR SEMICOLON

assign -> TVAR ASSIGN bitwise SEMICOLON

| AVAR ASSIGN bitwise SEMICOLON

| VVAR ASSIGN bitwise SEMICOLON

| SVAR ASSIGN bitwise SEMICOLON

assign -> TVAR L\_BRACKET arithmetic R\_BRACKET ASSIGN condition SEMICOLON

| AVAR L\_BRACKET arithmetic R\_BRACKET ASSIGN condition SEMICOLON

| VVAR L\_BRACKET arithmetic R\_BRACKET ASSIGN condition SEMICOLON

| SVAR L\_BRACKET arithmetic R\_BRACKET ASSIGN condition SEMICOLON

| SPVAR L\_BRACKET arithmetic R\_BRACKET ASSIGN condition SEMICOLON

| RAVAR L\_BRACKET arithmetic R\_BRACKET ASSIGN condition SEMICOLON

assign -> TVAR ASSIGN ABS L\_PAR var R\_PAR SEMICOLON

| AVAR ASSIGN ABS L\_PAR var R\_PAR SEMICOLON

| VVAR ASSIGN ABS L\_PAR var R\_PAR SEMICOLON

| SVAR ASSIGN ABS L\_PAR var R\_PAR SEMICOLON

bitwise -> var AND\_B var

| var OR\_B var

| var XOR\_B var

| var SHIFT\_L var

| var SHIFT\_R var

bitwise -> NOT\_B TVAR

| NOT\_B AVAR

| NOT\_B VVAR

| NOT\_B SVAR

conversion -> L\_PAR type R\_PAR var

type -> INT

| FLOAT

| CHAR

declaration -> var

declaration -> TVAR ASSIGN ARRAY L\_PAR R\_PAR SEMICOLON

| AVAR ASSIGN ARRAY L\_PAR R\_PAR SEMICOLON

| VVAR ASSIGN ARRAY L\_PAR R\_PAR SEMICOLON

| SVAR ASSIGN ARRAY L\_PAR R\_PAR SEMICOLON

| SPVAR ASSIGN ARRAY L\_PAR R\_PAR SEMICOLON

| RAVAR ASSIGN ARRAY L\_PAR R\_PAR SEMICOLON

declaration -> TVAR L\_BRACKET arithmetic R\_BRACKET SEMICOLON

| AVAR L\_BRACKET arithmetic R\_BRACKET SEMICOLON

| VVAR L\_BRACKET arithmetic R\_BRACKET SEMICOLON

| SVAR L\_BRACKET arithmetic R\_BRACKET SEMICOLON

| SPVAR L\_BRACKET arithmetic R\_BRACKET SEMICOLON

| RAVAR L\_BRACKET arithmetic R\_BRACKET SEMICOLON

empty -> epsilon