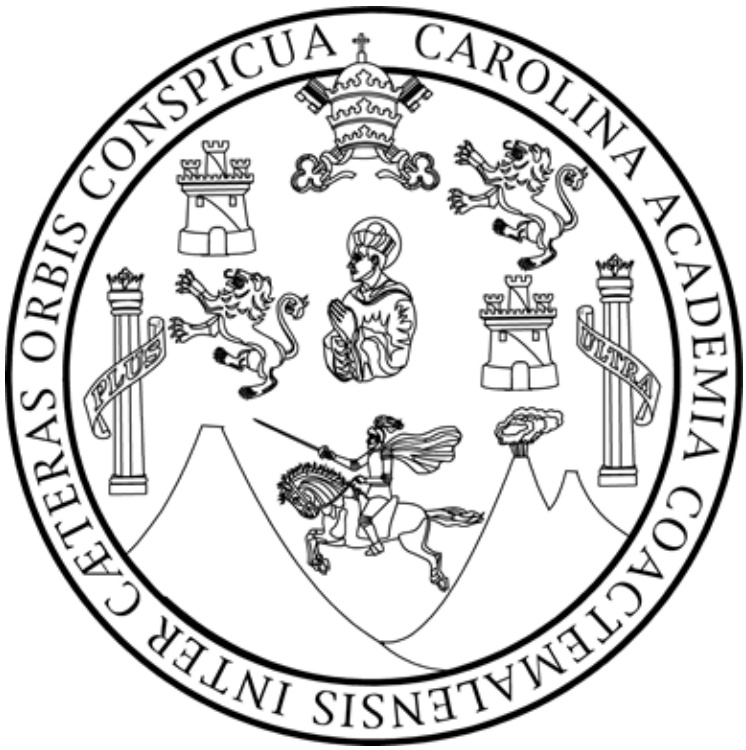
Universidad de San Carlos de Guatemala

Organización de lenguajes y compiladores 2

Sección A



Manual Técnico

Fernando Andrés Mérida Antón 201314713

17de Junio de 2020

Definición del proyecto

Augus es un lenguaje de programación, basado en PHP y en MIPS. Su principal funcionalidad es ser un lenguaje intermedio, ni de alto nivel como PHP ni de bajo nivel como el lenguaje ensamblador de MIPS.

El lenguaje tiene dos restricciones: la primera, es que cada instrucción es una operación simple; y la segunda, es que en cada instrucción hay un máximo de dos operandos y su asignación (si la hubiera).

Es un lenguaje débilmente tipado, sin embargo, si se reconocen cuatro tipos de datos no explícitos: entero, punto flotante, cadena de caracteres y arreglo.

Para manejar el flujo de control se proporciona la declaración de etiquetas, sin tener palabras reservadas para ese uso. Es decir, no hay ciclos for, while, ni do-while.

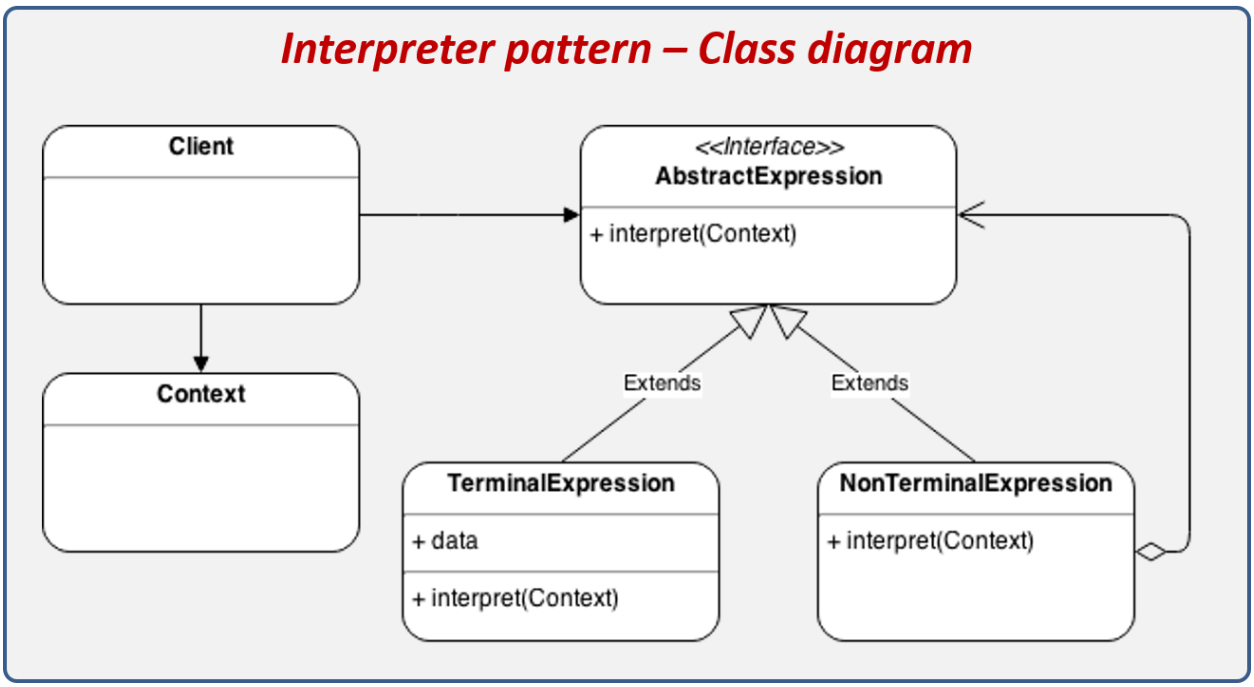
Entorno de desarrollo

- Python 3.8: Python es un lenguaje de programación interpretado, de alto nivel y de propósito general. Creado por Guido van Rossum y lanzado por primera vez en 1991, la filosofía de diseño de Python enfatiza la legibilidad del código con su notable uso de espacios en blanco significativos
- PLY 3.4: es una solución de Lex y Yacc para Python. Diseñada por It David M. Beazley.
- Tkinter: herramienta utilizada para crear la interfaz grafica

Descripción de código:

Se trabajo con un patrón interpreter para la solución de dicho problema en el que se utilizaron dos clases abstractas denominadas Expresión y Instrucción.

Patrón interpreter: En la programación de computadoras, el patrón de intérprete es un patrón de diseño que especifica cómo evaluar oraciones en un idioma. La idea básica es tener una clase para cada símbolo en un lenguaje informático especializado.



A continuación, se presentan estas dos clases

```
class Instruccion:
    '''This is an abstract class'''
    def ejecutar(self, ts,ms):
        pass
```

```
class Exp:

    def GetValor(self,ts,ms):
        pass
    def GetTipo(self,ts,ms):
        pass
```

Estas Clases a su vez son heredadas a cada una de las instrucciones y expresiones que se tienen dentro del proyecto.

Se realizo la definición de una tabla de símbolos que guarda los símbolos generados por el usuario con la siguiente estructura

```
from enum import Enum


class TIPO_DATO(Enum) :
```

```
    INTEGER = 1
    CHAR = 2
    FLOAT = 3
    BOOLEAN = 4
    ARRAY = 5
    STRUCT = 6

class Simbolo() :
    'Esta clase representa un simbolo dentro de nuestra tabla de simbolos'

    def __init__(self, id, tipo, valor,rol,dim,ambito,declarada) :
        self.id = id
        self.tipo = tipo
        self.valor = valor
        self.rol = rol
        self.reference = None
        self.dim= dim
        self.ambito = ambito
        self.declarada = declarada
        self.posicion = None

    def SetReference(self, id):
        self.reference = id

    def SetPosicion(self,id, accesos):
        if self.posicion is None:
            self.posicion = {}
            self.posicion[id] = accesos
        else:
            self.posicion[id] = accesos

class TablaDeSimbolos() :
    'Esta clase representa la tabla de simbolos'

    def __init__(self, simbolos = None) :
        if simbolos is None:
            self.simbolos = {}
        else:
            self.simbolos = simbolos

    def agregar(self, simbolo) :
        self.simbolos[simbolo.id] = simbolo

    def eliminar(self, id) :
        if not id in self.simbolos :
```

```
            return None

        self.simbolos.pop(id)

    def obtener(self, id) :
        if not id in self.simbolos :
            return None

        return self.simbolos[id]

    def actualizar(self, simbolo) :
        if not simbolo.id in self.simbolos :
            print('Error: variable ', simbolo.id, ' no definida.')
        else :
            self.simbolos[simbolo.id] = simbolo

    def printts(self) :
        for simbolo in self.simbolos:
            print("Simbolo: "+ str(self.simbolos[simbolo].id) + " Tipo: " +str(se
lf.simbolos[simbolo].tipo) +" Rol: "+str(self.simbolos[simbolo].rol)  +" Value:"
+str(self.simbolos[simbolo].valor))
```

Definición Léxica

reservadas = {

'int' : 'wint',

'char' : 'wchar',

'float': 'wfloat',

'goto' : 'wgoto',

'print': 'wprint',

'exit' : 'wexit',

'read' : 'wread',

'unset': 'wunset',

'abs' : 'wabs',

'xor' : 'wxor',

'if' : 'wif',

```
    'array'  : 'warray'
} tokens  = [
    'dollar',
    'ptocoma',
    'dosp',
    'parea',
    'parec',
    'corcha',
    'corchc',
    'igual',
    'mas',
    'menos',
    'por',
    'dividido',
    'modulo',
    'lnot',
    'land',
    'lor',
    'bnot',
    'band',
    'bor',
    'bxor',
    'bshiftl',
    'bshiftr',
    'menor',
    'mayor',
    'menorigual',
    'mayorigual',
    'igualdad',
```

```
    'diferente',
    'DECIMAL',
    'ENTERO',
    'CADENA',
    'CHAR',
    'ID'
```

Gramática ascendente

```
# Asociación de operadores y precedencia
  precedence = (
    ('right','igual'),
    ('left','lor'),
    ('left','land'),
    ('left','igualdad','diferente'),
    ('left','menor','mayor','menorigual','mayorigual'),
    ('left','mas','menos'),
    ('left','por','dividido'),
    ('right','Umenos'),
    ('left','parea','parec'),
    ) # Definición de la gramática

    INI     : LETS
    LETS    : LETS ET
    LETS    : ET
    ET      : ID dosp LINS
    LINS    : LINS INS
    LINS    : INS
    INS     : PRINT
            | ASIGNACION
```

| SALTO

| SIF

| EXIT

| UNSET

VAR : dollar ID LCOR

| dollar ID

LCOR : LCOR corcha EXP corchc

LCOR : corcha EXP corchc

PRINT : wprint parea EXP parec ptocoma

ASIGNACION : VAR igual EXP ptocoma

ASIGNACION : VAR igual band VAR ptocoma

SALTO : wgoto ID ptocoma

EXIT : wexit ptocoma

UNSET : wunset parea EXP parec ptocoma

SIF : wif parea EXP parec SALTO

EXP : EXP mas EXP

| EXP menos EXP

| EXP por EXP

| EXP dividido EXP

| EXP modulo EXP

|EXP mayor EXP

| EXP menor EXP

| EXP mayorigual EXP

| EXP menorigual EXP

| EXP igualdad EXP

| EXP diferente EXP

| EXP land EXP

| EXP lor EXP

| EXP wxor EXP

| lnot EXP

| EXP band EXP

| EXP bor EXP

| EXP bxor EXP

| EXP bshiftl EXP

| EXP bshiftr EXP

| bnot EXP

| menos EXP %prec Umenos

| parea EXP parec

| wabs parea EXP parec

| ENTERO

| DECIMAL

| VAR

| CADENA

| CHAR

| parea wint parec EXP

| parea wfloat parec EXP

| parea wchar parec EXP

| warray parea parec

| wread parea parec

Gramática descendente

Asociación de operadores y precedencia

precedence = (

```
    ('right','igual'),
    ('left','lor'),
    ('left','land'),
    ('left','bor'),
    ('left','wxor'),
    ('left','band'),
    ('left','igualdad','diferente'),
    ('left','menor','mayor','menorigual','mayorigual'),
    ('left','bshiftl','bshiftr'),
    ('left','mas','menos'),
    ('left','por','dividido','modulo'),
    ('right','Umenos','lnot','bnot'),
    ('left','parea','parec','corcha','corchc'),
    )

# Definición de la gramática

    INI     : LETS
    LETS    :  ET LETS
    LETS    : ET
    ET      : ID dosp LINS
    LINS    : INS LINS
    LINS    : INS
    INS     : PRINT
            | ASIGNACION
            | SALTO
            | SIF
            | EXIT
            | UNSET
```

VAR : dollar ID LCOR

VAR : dollar ID

LCOR : corcha EA corchc LCOR

LCOR : corcha EA corchc

PRINT : wprint parea EA parec ptocoma

ASIGNACION : VAR igual EA ptocoma

ASIGNACION : VAR igual band VAR ptocoma

SALTO : wgoto ID ptocoma

EXIT : wexit ptocoma

UNSET : wunset parea EA parec ptocoma

SIF : wif parea EA parec SALTO

EA : EB EAP

EAP : lor EB EAP

| empty

EB : EC EBP

EBP : land EC EBP

| empty

EC : ED ECP

ECP : bor ED ECP

| empty

ED : EE EDP

EDP : wxor EE EDP

| empty

EE : EF EEP

EEP : bxor EF EEP

| empty

EF : EG EFP

EFP : band EG EFP

| empty

EG : EH EGP

EGP : igualdad EH EGP

    | diferente EH EGP

    | empty

EH : EI EHP

EHP : mayor EI EHP

    | menor EI EHP

    | mayorigual EI EHP

    | menorigual EI EHP

    | empty

EI : EJ EIP

EIP : bshiftl EJ EIP

    | bshiftr EJ EIP

    | empty

EJ : EK EJP

EJP : mas EK EJP

    | menos EK EJP

    | empty

EK : E EKP

EKP : por E EKP

    | dividido E EKP

    | modulo E EKP

    | empty

E : lnot E

    | bnot E

    | menos E %prec Umenos

E : parea EA parec

E : wabs parea EA parec

E : ENTERO

E : DECIMAL

E : VAR

E : CADENA

E : CHAR

E: parea wint parec EA

| parea wfloat parec EA

| parea wchar parec EA

E : warray parea parec

E : wread parea parec

empty :