PROYECTO
AUGUS

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
Facultad de Ingeniería

MANUAL TECNICO

Luis Alfonso Ordoñez Carrillo
201603127

Proyecto No.1 Organización de lenguajes y compiladores 2

Contenido

Introducción

La finalidad de este manual es dar a conocer al lector la lógica con la que se desarrolló este software por medio de imágenes de las pantallas y explicaciones del codigo.

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.

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.

Objetivos

General

Proporcionar, al administrador del sistema u otros desarrolladores, una guía sobre las clases y sus atributos para facilitar la manipulación y el control del software.

Específicos

- Orientar al lector sobre la funcionalidad de los principales procesos del sistema.
- Facilitar el entendimiento del código para que el lector no tenga complicaciones al momento de realizar cambios en el programa.
- Dar a conocer la estructura de las gramáticas utilizadas para realizar el análisis ascendente y descendente
- Orientar al lector sobre el uso de graphviz y la creación de reportes en Python

Requisitos del sistema

Configuración mínima

- **Microsoft Windows Vista SP1 / Windows 7 Professional:**
  - **Procesador:** Intel Pentium III a 800 MHz o equivalente
  - **Memoria:** 512 MB
  - **Espacio en disco:** 750 MB de espacio libre en disco
- **Ubuntu 9.10:**
  - **Procesador:** Intel Pentium III a 800 MHz o equivalente
  - **Memoria:** 512 MB
  - **Espacio en disco:** 650 MB de espacio libre en disco
- **Macintosh OS X 10.7 Intel:**
  - **Procesador:** Intel Dual-Core
  - **Memoria:** 2 GB
  - **Espacio en disco:** 650 MB de espacio libre en disco

Configuración recomendada

- **Microsoft Windows 7 Professional / Windows 8 / Windows 8.2:**
  - **Procesador:** Intel Core i5 o equivalente
  - **Memoria:** 2 GB (32 bits), 4GB (64bits)
  - **Espacio en disco:** el espacio libre en disco de 1,5 GB
- **Ubuntu 15.04:**
  - **Procesador:** Intel Core i5 o equivalente
  - **Memoria:** 2 GB (32 bits), 4GB (64bits)
  - **Espacio en disco:** el espacio libre en disco de 1,5 GB
- **OS X 10.10 Intel:**
  - **Procesador:** Intel Dual-Core
  - **Memoria:** 4 GB
  - **Espacio en disco:** el espacio libre en disco de 1,5 GB

GRAMATICA ASCENDENTE

```
inicio : instrucciones

instrucciones : instrucciones instruccion
              | instruccion

instruccion : etiqueta_main
            | etiqueta_ID
            | asignacion
            | inst_goto
            | inst_if
            | inst_print
            | inst_exit
            | inst_unset

etiqueta_main : MAIN DOSPUNTOS

etiqueta_ID : ID DOSPUNTOS

asignacion : variables IGUAL operaciones PUNTOCOMA

asignacion : variables_array IGUAL operaciones PUNTOCOMA

inst_goto : GOTO ID PUNTOCOMA

inst_if : IF PARENTA operaciones PARENTC GOTO ID PUNTOCOMA

inst_print : PRINT PARENTA variables PARENTC PUNTOCOMA
           | PRINT PARENTA val PARENTC PUNTOCOMA

inst_unset : UNSET PARENTA variables PARENTC PUNTOCOMA

inst_exit : EXIT PUNTOCOMA

operaciones : val MAS val
            | val MENOS val
            | val POR val
            | val DIVICION val
            | val RESIDUO val
            | val AND1 val
            | val OR1 val
            | val XOR val
            | val AND2 val
            | val OR2 val
            | val XOR2 val
            | val SHIFTI val
            | val SHIFTD val
            | val IGUALIGUAL val
            | val DIFERENTE val
            | val MAYORIGUAL val
            | val MENORIGUAL val
            | val MAYOR val
            | val MENOR val
            | MENOS val
            | EXCLAMA val
            | NOT val
            | ABS PARENTA val PARENTC
            | READ PARENTA PARENTC
            | ARRAY PARENTA PARENTC
            | val

val : conversiones
    | ENTERO
    | DECIMAL
    | CADENA
    | variables
    | variables_array

variables : TEMPORALES
          | PARAMETROS
          | VALORES_DEVUELTOS
          | SIMULADO
          | PILA
          | PUNTERO_PILA

variables_array : TEMPORALES indices
                | PARAMETROS indices
                | VALORES_DEVUELTOS indices
                | SIMULADO indices
                | PILA indices
                | PUNTERO_PILA indices

indices : indices indice
        | indice

indice : CORCHEA val CORCHEC

conversiones : PARENTA INT PARENTC val
             | PARENTA FLOAT PARENTC val
             | PARENTA CHAR PARENTC val
```

GRAMATICA DESCENDENTE

```
inicio: instrucciones

instrucciones: instruccion instrucciones_p

instrucciones_p: instruccion instrucciones_p
               |3

instruccion: etiqueta_main
           | etiqueta_ID
           | asignacion
           | inst_goto
           | inst_if
           | inst_print
           | inst_exit
           | inst_unset

etiqueta_main: MAIN DOSPUNTOS

etiqueta_ID: ID DOSPUNTOS

asignacion: variables IGUAL operaciones PUNTOCOMA
          | variables_array IGUAL operaciones PUNTOCOMA

inst_goto : GOTO ID PUNTOCOMA

inst_if : IF PARENTA operaciones PARENTC GOTO ID PUNTOCOMA

inst_print : PRINT PARENTA variables PARENTC PUNTOCOMA
           | PRINT PARENTA val PARENTC PUNTOCOMA

inst_unset : UNSET PARENTA variables PARENTC PUNTOCOMA

inst_exit : EXIT PUNTOCOMA

operaciones : val operaciones_p

operaciones_p :  MAS val
                | MENOS val
                | POR val
                | DIVICION val
                | RESIDUO val
                | AND1 val
                | OR1 val
                | XOR val
                | AND2 val
                | OR2 val
                | XOR2 val
                | SHIFTI val
                | SHIFTD val
                | IGUALIGUAL val
                | DIFERENTE val
                | MAYORIGUAL val
                | MENORIGUAL val
                | MAYOR val
                | MENOR val

operaciones : MENOS val
            | EXCLAMA val
            | NOT val
            | ABS PARENTA val PARENTC
            | READ PARENTA PARENTC
            | ARRAY PARENTA PARENTC

operaciones : val

val: conversiones
   | ENTERO
   | DECIMAL
   | CADENA
   | variables
   | variables_array

variables : TEMPORALES
          | PARAMETROS
          | VALORES_DEVUELTOS
          | SIMULADO
          | PILA
          | PUNTERO_PILA

variables_array : TEMPORALES indices
                | PARAMETROS indices
                | VALORES_DEVUELTOS indices
                | SIMULADO indices
                | PILA indices
                | PUNTERO_PILA indices

indices : indice indices_P

indices_P : indice indices_P
          | 3

indice : CORCHEA val CORCHEC

conversiones : PARENTA INT PARENTC val
             | PARENTA FLOAT PARENTC val
             | PARENTA CHAR PARENTC val
```

Clase Main

En la clase Main, es donde inicia la aplicación.

Primero se crea una clase llamda Ui_MainWindow que es la que va a recibir todos nuestros objetos de QWidgets que formaran parte de nuestra venta principal

Mas abajo también se declaran acciones para cada menú que necesitaremos para realizar diferentes acciones como ejecutar, debuggear o generar reportes

Tambien cuenta con otras funciones que nos permiten manejar los archivos y las pestañas.

```
29  class Ui_MainWindow(object):
30      array_rutas = []
31      array_editores = []
32      array_nombre_tab = []
33      nombre_archivo = ""
34      ruta_archivo = ""
35      Fuente = QtGui.QFont("Microsoft Sans Serif",10)
36      Fuente2 = QtGui.QFont("Ebrima", 12)
37      ruta_iconos = str(os.getcwd()) + "\Iconos"
38      tabla_global = None
39      indice = -1
40
41      def setupUi(self, MainWindow):
42          MainWindow.setObjectName("MainWindow")
43          MainWindow.setWindowTitle("MI PROGRAMOTA!!")
44          MyIcon = QtGui.QIcon(self.ruta_iconos + "\\favicon.png")
45          MainWindow.setWindowIcon(MyIcon)
46          MainWindow.resize(1355, 768)
47          self.centralwidget = QtWidgets.QWidget(MainWindow)
48          self.centralwidget.setObjectName("centralwidget")
49
50          self.tbTab = QtWidgets.QTabWidget(self.centralwidget)
51          self.tbTab.setGeometry(QtCore.QRect(20, 10, 770, 490))
52          self.tbTab.setObjectName("tbTab")
53
54          #Declaracion de la consola
55          self.groupBox = QtWidgets.QGroupBox(self.centralwidget)
56          self.groupBox.setGeometry(QtCore.QRect(20, 510, 770, 190))
57          self.groupBox.setObjectName("groupBox")
58          self.groupBox.setFont(self.Fuente)
59          self.txtConsola = QtWidgets.QPlainTextEdit(self.groupBox)
60          self.txtConsola.setGeometry(QtCore.QRect(10, 20, 750, 160))
61          self.txtConsola.setObjectName("txtConsola")
62          self.txtConsola.setFont(self.Fuente2)
63          self.txtConsola.setStyleSheet("""QPlainTextEdit {background-color: #333; color: #00FF00;}""")
64          self.txtConsola.setReadOnly(True)
65
66
67          #Creacion del visor de imagenes
68          self.visor = QtWidgets.QGraphicsView(self.centralwidget)
69          self.visor.setGeometry(QtCore.QRect(795, 10, 555, 690))
70          self.visor.setObjectName("qgVisor")
71
72          MainWindow.setCentralWidget(self.centralwidget)
73
```

Clase Editor de código

Esta es una clase que se diseñó específicamente para poder colocar numero al editor de texto y que este mismo se vaya actualizando con forme se va escribiendo, también cuenta con la funcionalidad de subrayar la línea en la que estamos trabajando con un ligero color celeste.

Esta clase hereda todos los métodos de un QPlaintexEdit normal, lo que nos permite usar los métodos y funciones propios del mismo

```
from PyQt5.QtCore import Qt, QRect, QSize
from PyQt5.QtWidgets import QWidget, QPlainTextEdit, QTextEdit
from PyQt5.QtGui import QColor, QPainter, QTextFormat


class QLineNumberArea(QWidget):
    def __init__(self, editor):
        super().__init__(editor)
        self.codeEditor = editor

    def sizeHint(self):
        return QSize(self.editor.lineNumberAreaWidth(), 0)

    def paintEvent(self, event):
        self.codeEditor.lineNumberAreaPaintEvent(event)


class QCodeEditor(QPlainTextEdit):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.lineNumberArea = QLineNumberArea(self)
        self.blockCountChanged.connect(self.updateLineNumberAreaWidth)
        self.updateRequest.connect(self.updateLineNumberArea)
        self.cursorPositionChanged.connect(self.highlightCurrentLine)
        self.updateLineNumberAreaWidth(0)

    def lineNumberAreaWidth(self):
        digits = 1
        max_value = max(1, self.blockCount())
        while max_value >= 10:
            max_value /= 10
            digits += 1
        space = 3 + self.fontMetrics().width('9') * digits
        return space

    def updateLineNumberAreaWidth(self, _):
        self.setViewportMargins(self.lineNumberAreaWidth(), 0, 0, 0)

    def updateLineNumberArea(self, rect, dy):
        if dy:
            self.lineNumberArea.scroll(0, dy)
        else:
            self.lineNumberArea.update(0, rect.y(), self.lineNumberArea.width(), rect.height())
        if rect.contains(self.viewport().rect()):
            self.updateLineNumberAreaWidth(0)
```

Clase Valores_Funciones

Dentro de este documento se encuentran todas las estructuras de las variables que se manejan int, double, strings, también se guardan las operaciones binarias y unarias.

También se crean varios enum para ayudarnos a hacer mas fácil el análisis de que tipo de operación es cada uno y que tipo de datos hay en cada operación

Cada clase también, guarda el no de fila y columna donde fue encontrado, esto con fines de hacer mas fácil el análisis de los errores semánticos

```
class Tipo_Variable(Enum):
    TEMPORALES = 1
    PARAMETROS = 2
    DEVUELTOS = 3
    PILA = 4
    SIMULADOR = 5
    PUNTERO_PILA = 6

class Operacion_Convercion(Enum):
    TO_INT = 1
    TO_STR = 2
    TO_CHAR = 3


class Val_Numerico:
    '''print("clase val_num")'''

class Operacion_Binaria():
    def __init__(self, val1, val2, operacion, fila, columna):
        self.val1 = val1
        self.val2 = val2
        self.operacion = operacion
        self.fila = fila
        self.columna = columna

class Numerico_Negativo(Val_Numerico):
    def __init__(self, val, fila, columna):
        self.val = val
        self.fila = fila
        self.columna = columna

class Numerico_Absoluto(Val_Numerico):
    def __init__(self, val, fila, columna):
        self.val = val
        self.fila = fila
        self.columna = columna

#PRIMITIVOS
class Numerico_Entero(Val_Numerico):
    def __init__(self, val, tipo, fila, columna):
        self.val = val
        self.tipo = tipo
        self.fila = fila
        self.columna = columna
```

Clase Métodos

Dentro de esta clase se encuentran varias estructuras que nos ayudaran a la hora de ejecutar todas las instrucciones, generadas por el árbol. Estas también nos ayudaran con la generación del AST

```python
class Metodos:
    '''clase  abstracta'''

class Asignacion(Metodos):
    def __init__(self, variable, valor, fila, columna):
        self.variable = variable
        self.valor = valor
        self.fila = fila
        self.columna = columna

class Imprimir(Metodos):
    def __init__(self, mensaje, fila, columna):
        self.mensaje = mensaje
        self.fila = fila
        self.columna = columna

class If_Goto(Metodos):
    def __init__(self, op_logica, goto_etiqueta, fila, columna):
        self.op_logica = op_logica
        self.goto_etiqueta = goto_etiqueta
        self.fila = fila
        self.columna = columna

class Goto(Metodos):
    def __init__(self, etiqueta, fila, columna):
        self. etiqueta = etiqueta
        self.fila = fila
        self.columna = columna

class Read(Metodos):
    def __init__(self, fila, columna):
        '''Esta clase es del read'''
        self.fila = fila
        self.columna = columna

class Unset(Metodos):
    def __init__(self, variable, fila, columna):
        self.variable = variable
        self.fila = fila
        self.columna = columna

class Exit(Metodos):
    def __init__(self, fila, columna):
        ''''Esta clase es del exit'''
        self.fila = fila
        self.columna = columna
```

Tabla de Simbolos

En el documento de clases de simbolos podemos encontrar la clase símbolo que es el nodo en donde vamos a estar almacenando todos los valores que sean validos de la ejecucion y vamos a tener una clase llamada tabla_simbolos que tiene sus métodos de get y set para hacer un poco mas el guardado y el sacado de los simbolos de la misma

```
class Simbolo():
    def __init__(self, id, tipo, valor, dimension, declarada, referencias):
        self.id = id
        self.tipo = tipo
        self.valor = valor
        self.dimension = dimension
        self. declarada = declarada
        self.referencias = referencias

class Tabla_Simbolos():
    def __init__(self, simbolos = {}):
        self.simbolos = simbolos

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

    def get_simbolo(self, id):
        if not id in self.simbolos:
            print("no existe el simbolo")
            return Simbolo(None,None,None,None,None,None)
        return self.simbolos[id]
        #imprimir error

    def update_simbolo(self, simbolo):
        if simbolo.id in self.simbolos:
            self.simbolos[simbolo.id] = simbolo

            #imprimir error

    def existe_simbolo(self, simbolo):
        if simbolo.id in self.simbolos:
            return True
        return False

    def existe_id(self, id):
        if id in self.simbolos:
            return True
        return False

    def clear(self):
        self.simbolos.clear()

    def get_all(self):
        return self.simbolos

    def delete_simbolo(self, id):
        if id in self.simbolos:
            texto = self.simbolos.pop(id)
```

Clase Errores

En esta tenemos una lista llamada Lista_errores que se declara como vacia y con visibilidad publica, lo que nos facilitara la inserción de los errores donde sea que los necesitemos. Tambien tenemos la clase Error que es el nodo donde se guarda lo necesario para hacer el reporte de errores luego

```
1  Lista_errores = []
2
3  class Error():
4      def __init__(self, token, tipo, desc, fila, columna):
5          self.token = token
6          self.tipo = tipo
7          self.desc = desc
8          self.fila = fila
9          self.columna = columna
```