Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Lenguajes Formales y de Programación

Sección B+

Fecha: 19/08/2020


# MANUAL TÉCNICO

Nombre: Carné:

Gerson Rubén Quiroa del Cid 2020 00166

# Índice

# Requisitos del sistema

Para la instalación de la aplicación, su computadora y/o laptop debe cumplir como mínimo los siguientes requerimientos:

## Para Windows

- Windows Vista SP2 (8u51 y superiores)
- Windows Server 2008 R2 SP1 (64 bits)
- Windows Server 2012 y 2012 R2 (64 bits)
- RAM: 128 MB
- Procesador: Mínimo Pentium 2 a 266 MHz
- Exploradores: Internet Explorer 9 y superior, Firefox

## Mac OS

- Mac con Intel que ejecuta Mac OS X 10.8.3+, 10.9+
- Privilegios de administrador para la instalación
- Explorador de 64 bits
- Se requiere un explorador

## Linux

- Oracle Linux 7.x (64 bits)2(8u20 y superiores)
- Red Hat Enterprise Linux 7.x (64 bits)2(8u20 y superiores)
- Suse Linux Enterprise Server 12.x (64 bits)2(8u31 y superiores)
- Ubuntu Linux 12.04 LTS, 13.x
- Ubuntu Linux 15.10 (8u65 y superiores)

**IMPORTANTE:** Independientemente del sistema operativo del usuario, es necesario que previamente instale Python en su computadora.

**Editor utilizado:** Visual Studio Code IDE 8.2

**Versión de Python utilizado:** 3.9.4

# Introducción

El manual técnico tiene como finalidad de explicar el funcionamiento de cada función utilizado en el presente programa, con el objetivo de que no haya confusión al leer el código del programa y cualquier persona con conocimiento básico de programación en Python pueda comprender lo que se hizo en cada línea de código. Por lo cual se recomienda que toda persona que quiera leer o modificar el código vea este manual para ahorraste tiempo y así sea más fácil su comprensión.

## Interfaz gráfica

Estas clases son las que conforman la interfaz gráfica, cada clase representa una ventana y éstas son las siguientes:

### Clase GUI (Main)

Esta clase es la principal del programa donde inicialmente se ejecutará el menú con las diferentes opciones que puede elegir el usuario, dependiendo de la opción que se elija el usuario se ejecutarán las diferentes funciones detalladas más adelante.

```python
import sys
from PyQt5 import uic
from PyQt5.QtWidgets import QMainWindow, QApplication, QFileDialog, QDialog
from Analizador import analizar, tokens, errores, img
from HTML import reportes
from PyQt5.QtGui import QPixmap
from PyQt5.QtCore import QSize, Qt

class GUI(QMainWindow):
    def __init__(self):
        super().__init__()
        self.contenido = None
        self.errorDialog = ErrorDialog()
        self.mensajeExito = MensajeExito()
        uic.loadUi("GUI/interfaz.ui", self)
        self.label_2.setHidden(True)
        self.label_3.setHidden(True)
        # botones clickeados y por parámetro las funciones que desencadenan
        self.botonCargar.clicked.connect(self.browsefiles)
        self.botonAnalizar.clicked.connect(self.analizarUI)
        self.botonReporte.clicked.connect(self.reportes)
        self.botonActualizar.clicked.connect(self.actualizar)
        self.botonOriginal.clicked.connect(self.original)
        self.botonMX.clicked.connect(self.mirrorx)
        self.botonMY.clicked.connect(self.mirrory)
        self.botonDM.clicked.connect(self.doublemirror)
        self.botonSalir.clicked.connect(exit)
```

### Clase ErrorDialog

Esta clase es la ventana de error, se abre cuando no se reconocen los archivos o no se puede ingresar a cierta opción.

```
88  class ErrorDialog(QDialog):
89      def __init__(self):
90          super().__init__()
91          uic.loadUi("GUI/errorDialog.ui", self)
92          # botones clickeados y por parámetro las funciones que desencadenan
93          self.botonAceptar.clicked.connect(self.close)
```

### Clase MensajeExito

Esta clase es la ventana que se abre cuando se ha ingresado correctamente los archivos.

```
class MensajeExito(QDialog):
    def __init__(self):
        super().__init__()
        uic.loadUi("GUI/mensajeExito.ui", self)
        # botones clickeados y por parámetro las funciones que desencadenan
        self.botonAceptar.clicked.connect(self.close)
```

### Variables globales

Las diferentes variables globales se utilizaron para manejar algunos datos de mejor manera, estas variables son las siguientes:

```
105    app = QApplication(sys.argv)
106    gui = GUI()
107    gui.show()
108    sys.exit(app.exec_())
```

```
6    # lista de tokens y errores
7    tokens = []
8    errores = []
```

```
216    # lista de imagenes
217    img = []
```

## Clase Analizador

Esta clase es la encargada de analizar el archivo de entrada mediante un autómata finito determinista definido, así como también se encarga de guardar toda la información en objetos descritos posteriormente.

## Función analizar:

Esta función es la que contiene el autómata que reconoce cada carácter que haya dentro del archivo de entrada, si no encuentra el carácter, lo mandará a una lista de errores.

```
10  def analizar(contenido):
11      # inicializando variables
12      linea = 1
13      columna = 1
14      buffer = ""
15      centinela = "$"
16      estado = 0
17      i = 0
18      contenido += centinela
19
20      # autómata
21      while(i < len(contenido)):
22          # leyendo caracter por caracter
23          c = contenido[i]
24
25          # estado 0
26          if estado == 0:
27              if c == "=": # signo igual
28                  buffer += c
29                  tokens.append(Token(buffer, "Signo igual", linea, columna))
30                  buffer = ""
31                  columna += 1
32              elif c == "{": # llave de apertura
33                  buffer += c
34                  tokens.append(Token(buffer, "Llave de apertura", linea, columna))
35                  buffer = ""
36                  columna += 1
37              elif c == "}": # llave de cierre
38                  buffer += c
39                  tokens.append(Token(buffer, "Llave de cierre", linea, columna))
40                  buffer = ""
```

## Función guardarDatos

Esta función es la que se encarga de guardar todos los datos necesarios en una lista de objetos de tipo Imagen, para generar las imágenes.

```
218  def guardarDatos(tokens):
219      titulo = ""
220      ancho = ""
221      alto = ""
222      filas = ""
223      columnas = ""
224      x = ""
225      y = ""
226      flag = ""
227      hexadecimal = ""
228      celdas = []
229      filtros = []
230      i = 1
231      for t in tokens:
232          if t.tipo == "Separador" or i == len(tokens):
233              # agregando la imagen a la lista de imagenes
234              img.append(Imagen(titulo, ancho, alto, filas, columnas, celdas, filtros))
235              titulo = ""
236              ancho = ""
237              alto = ""
238              filas = ""
239              columnas = ""
240              celdas = []
241              filtros = []
242
243          if t.tipo == "Corchete de cierre":
244              celdas.append(Celda(x, y, flag, hexadecimal))
245              x = ""
246              y = ""
247              flag = ""
248              hexadecimal = ""
```

## Clase HTML

Esta clase, es en donde se encuentras todos los archivos html que se generarán a lo largo de la ejecución del programa, como pueden ser los reportes de tokens y errores y así como las imágenes en html.

### Función repTokens

Esta función es la que genera el reporte de tokens.

```python
def repTokens(tokens):
        i = 1
        html = open("Reportes/Tokens.html", 'w')
        html.write('<!DOCTYPE html><!--Declarar el tipo de cumento -->'
                + '<html>'
                + '<!--Encabezado-->'
                + '<head>'
                + '<meta charset="UTF-8"><!--codifiifcaion de caracteres ñ y á-->'
                + '<meta name="name" content="Reporte"><!--nombre de la pagina-->'
                + '<meta name="description" content="name"><!--autor de la pagina-->'
                + '<meta name="keywods" content="uno,dos,tres"><!--Palabras claavez para, separadas por comas--
                + '<meta name="robots" content="Index, Follow"><!--Mejora la busqueda-->'
                + '<meta name="viewport" content="width=device-width, initial-scale=1"><!--visibilidaad en dife
                + '<link rel="stylesheet" type="text/css" href="css/styles.css"/><!--css /estilo/tipo/ruta rela
                + '<title>Tokens</title><!--Titulo visible de la pagina-->'
                + '</head>'
                + '<body>'
                + '<center><!--centra todos lo que este dentro-->'
                + '<h6 class="titulos"><b> TOKENS </b></h6>')
        html.write('<br>  <br>  <br>'
                + '<!----tabla 2-->'
                + '<table class="steelBlueCols">'
                + '<thead>'
                + '<tr> <th>No.</th> <th>Token</th> <th>Lexema</th> <th>Fila</th> <th>Columna</th> </tr>'
                + '</thead>'
                + '<tbody>')
        for t in tokens:
                html.write(f'<tr> <td>{str(i)}</td> <td>{t.tipo}</td> <td>{t.lexema}</td> <td>{str(t.linea)}</t
                i += 1
        html.write('</tbody>'
                + '</table>'
```

## Función repErrores

Esta función es la que se encarga de crear el reporte con los errores que contenga el archivo de entrada.

```python
def repErrores(errores):
        i = 1
        html = open("Reportes/Errores.html", 'w')
        html.write('<!DOCTYPE html><!--Declarar el tipo de cumento -->'
                + '<html>'
                + '<!--Encabezado-->'
                + '<head>'
                + '<meta charset="UTF-8"><!--codifiifcaion de caracteres ñ y á-->'
                + '<meta name="name" content="Reporte"><!--nombre de la pagina-->'
                + '<meta name="description" content="name"><!--autor de la pagina-->'
                + '<meta name="keywods" content="uno,dos,tres"><!--Palabras claavez para, separadas por comas--
                + '<meta name="robots" content="Index, Follow"><!--Mejora la busqueda-->'
                + '<meta name="viewport" content="width=device-width, initial-scale=1"><!--visibilidaad en dife
                + '<link rel="stylesheet" type="text/css" href="css/styles.css"/><!--css /estilo/tipo/ruta rela
                + '<title>Errores</title><!--Titulo visible de la pagina-->'
                + '</head>'
                + '<body>'
                + '<center><!--centra todos lo que este dentro-->'
                + '<h6 class="titulos"><b> ERRORES </b></h6>')
        html.write('<br>  <br>  <br>'
                + '<!----tabla 2-->'
                + '<table class="steelBlueCols">'
                + '<thead>'
                + '<tr> <th>No.</th> <th>Fila</th> <th>Columna</th> <th>Caracter</th></tr>'
                + '</thead>'
                + '<tbody>')
        for e in errores:
                html.write(f'<tr> <td>{str(i)}</td> <td>{str(e.linea)}</td> <td>{str(e.columna)}</td> <td>{e.de
                i += 1
        html.write('</tbody>'
                + '</table>'
```

## Función original

Esta función es la que genera la imagen que contiene el archivo de entrada sin ninguna modificación.

```
def original(img):
        for i in img:
                html = open(f"Imagenes/Original/{i.titulo}.html", 'w')
                html.write('<!DOCTYPE html><!--Declarar el tipo de cumento -->'
                        + '<html>'
                        + '<!--Encabezado-->'
                        + '<head>'
                        + '<meta charset="UTF-8"><!--codififcaion de caracteres ñ y á-->'
                        + '<meta name="name" content="Reporte"><!--nombre de la pagina-->'
                        + '<meta name="description" content="name"><!--autor de la pagina-->'
                        + '<meta name="keywods" content="uno,dos,tres"><!--Palabras claavez para, separadas por
                        + '<meta name="robots" content="Index, Follow"><!--Mejora la busqueda-->'
                        + '<meta name="viewport" content="width=device-width, initial-scale=1"><!--visibilidaad
                        + f'<title>{i.titulo}</title><!--Titulo visible de la pagina-->'
                        + '</head>'
                        + '<body>'
                        + '<center><!--centra todos lo que este dentro-->'
                        + f'<h1 class="titulos"><b> {i.titulo} </b></h1>')
                html.write('<br>  <br>'
                        + '<table border cellspacing=0 class="default">')

                f = 1
                for filas in range(i.filas):
                        c = 1
                        html.write('<tr>')
                        for columnas in range(i.columnas):
                                flag = True
                                for celda in i.celdas:
                                        # si está en la lista de celdas se pintará, de lo contrario no
                                        # width = ancho, height = alto
```

## Función mirrorx

Esta función genera la misma imagen original pero rotada horizontalmente.

```
def mirrorx(img):
        for i in img:
                html = open(f"Imagenes/MirrorX/{i.titulo}.html", 'w')
                html.write('<!DOCTYPE html><!--Declarar el tipo de cumento -->'
                        + '<html>'
                        + '<!--Encabezado-->'
                        + '<head>'
                        + '<meta charset="UTF-8"><!--codififcaion de caracteres ñ y á-->'
                        + '<meta name="name" content="Reporte"><!--nombre de la pagina-->'
                        + '<meta name="description" content="name"><!--autor de la pagina-->'
                        + '<meta name="keywods" content="uno,dos,tres"><!--Palabras claavez para, separadas por
                        + '<meta name="robots" content="Index, Follow"><!--Mejora la busqueda-->'
                        + '<meta name="viewport" content="width=device-width, initial-scale=1"><!--visibilidaad
                        + f'<title>{i.titulo}</title><!--Titulo visible de la pagina-->'
                        + '</head>'
                        + '<body>'
                        + '<center><!--centra todos lo que este dentro-->'
                        + f'<h1 class="titulos"><b> {i.titulo} </b></h1>')
                html.write('<br>  <br>'
                        + '<table border cellspacing=0 class="default">')

                f = 1
                for filas in range(i.filas):
                        c = i.columnas
                        html.write('<tr>')
                        for columnas in range(i.columnas, 1, -1):
                                flag = True
                                for celda in i.celdas:
```

### Función mirrory

Esta función genera la misma imagen original pero rotada verticalmente.

```
def mirrory(img):
        for i in img:
                html = open(f"Imagenes/MirrorY/{i.titulo}.html", 'w')
                html.write('<!DOCTYPE html><!--Declarar el tipo de cumento -->'
                        + '<html>'
                        + '<!--Encabezado-->'
                        + '<head>'
                        + '<meta charset="UTF-8"><!--codififcaion de caracteres ñ y á-->'
                        + '<meta name="name" content="Reporte"><!--nombre de la pagina-->'
                        + '<meta name="description" content="name"><!--autor de la pagina-->'
                        + '<meta name="keywods" content="uno,dos,tres"><!--Palabras claavez para, separadas por
                        + '<meta name="robots" content="Index, Follow"><!--Mejora la busqueda-->'
                        + '<meta name="viewport" content="width=device-width, initial-scale=1"><!--visibilidaad
                        + f'<title>{i.titulo}</title><!--Titulo visible de la pagina-->'
                        + '</head>'
                        + '<body>'
                        + '<center><!--centra todos lo que este dentro-->'
                        + f'<h1 class="titulos"><b> {i.titulo} </b></h1>')
                html.write('<br>  <br>'
                        + '<table border cellspacing=0 class="default">')

                f = i.filas
                for filas in range(i.filas, 1, -1):
                        c = 1
                        html.write('<tr>')
                        for columnas in range(i.columnas):
```

### Función doblemirror

Esta función genera la misma imagen original pero rotada tanto horizontalmente como verticalmente.

```
def doblemirror(img):
        for i in img:
                html = open(f"Imagenes/DoubleMirror/{i.titulo}.html", 'w')
                html.write('<!DOCTYPE html><!--Declarar el tipo de cumento -->'
                        + '<html>'
                        + '<!--Encabezado-->'
                        + '<head>'
                        + '<meta charset="UTF-8"><!--codififcaion de caracteres ñ y á-->'
                        + '<meta name="name" content="Reporte"><!--nombre de la pagina-->'
                        + '<meta name="description" content="name"><!--autor de la pagina-->'
                        + '<meta name="keywods" content="uno,dos,tres"><!--Palabras claavez para, separadas por
                        + '<meta name="robots" content="Index, Follow"><!--Mejora la busqueda-->'
                        + '<meta name="viewport" content="width=device-width, initial-scale=1"><!--visibilidaad
                        + f'<title>{i.titulo}</title><!--Titulo visible de la pagina-->'
                        + '</head>'
                        + '<body>'
                        + '<center><!--centra todos lo que este dentro-->'
                        + f'<h1 class="titulos"><b> {i.titulo} </b></h1>')
                html.write('<br>  <br>'
                        + '<table border cellspacing=0 class="default">')

                f = i.filas
                for filas in range(i.filas, 1, -1):
                        c = i.columnas
                        html.write('<tr>')
                        for columnas in range(i.columnas, 1, -1):
```

## Clases Objetos

Estas clases se utilizaron para crear diferentes objetos que se utilizaron en el programa, estas clases son las siguientes:

### Clase Token

Esta clase contiene los diferentes tokens.

```python
class Token(object):
    def __init__(self, lexema, tipo, linea, columna):
        self.lexema = lexema
        self.tipo = tipo
        self.linea = linea
        self.columna = columna

    def __str__(self):
        cadena = self.lexema + "; " + self.tipo + "; " + str(self.linea) + "; " + str(self.columna)
        return cadena
```

## Clase Error

Esta clase contiene los diferentes errores.

```python
class Error(object):
    def __init__(self, descrip, tipo, linea, columna):
        self.descrip = descrip
        self.tipo = tipo
        self.linea = linea
        self.columna = columna

    def __str__(self):
        cadena = self.descrip + "; " + self.tipo + "; " + str(self.linea) + "; " + str(self.columna)
        return cadena
```

## Clase Imagen

Esta clase contiene los datos de las imágenes.

```python
class Imagen(object):
    def __init__(self, titulo, ancho, alto, filas, columnas, celdas, filtros):
        self.titulo = titulo
        self.ancho = ancho
        self.alto = alto
        self.filas = filas
        self.columnas = columnas
        self.celdas = celdas
        self.filtros = filtros
```

## Clase Celda

Esta clase contiene las diferentes celdas para que se genere las imágenes.

```python
class Celda(object):
    def __init__(self, x, y, activo, color):
        self.x = x
        self.y = y
        self.activo = activo
        self.color = color
```

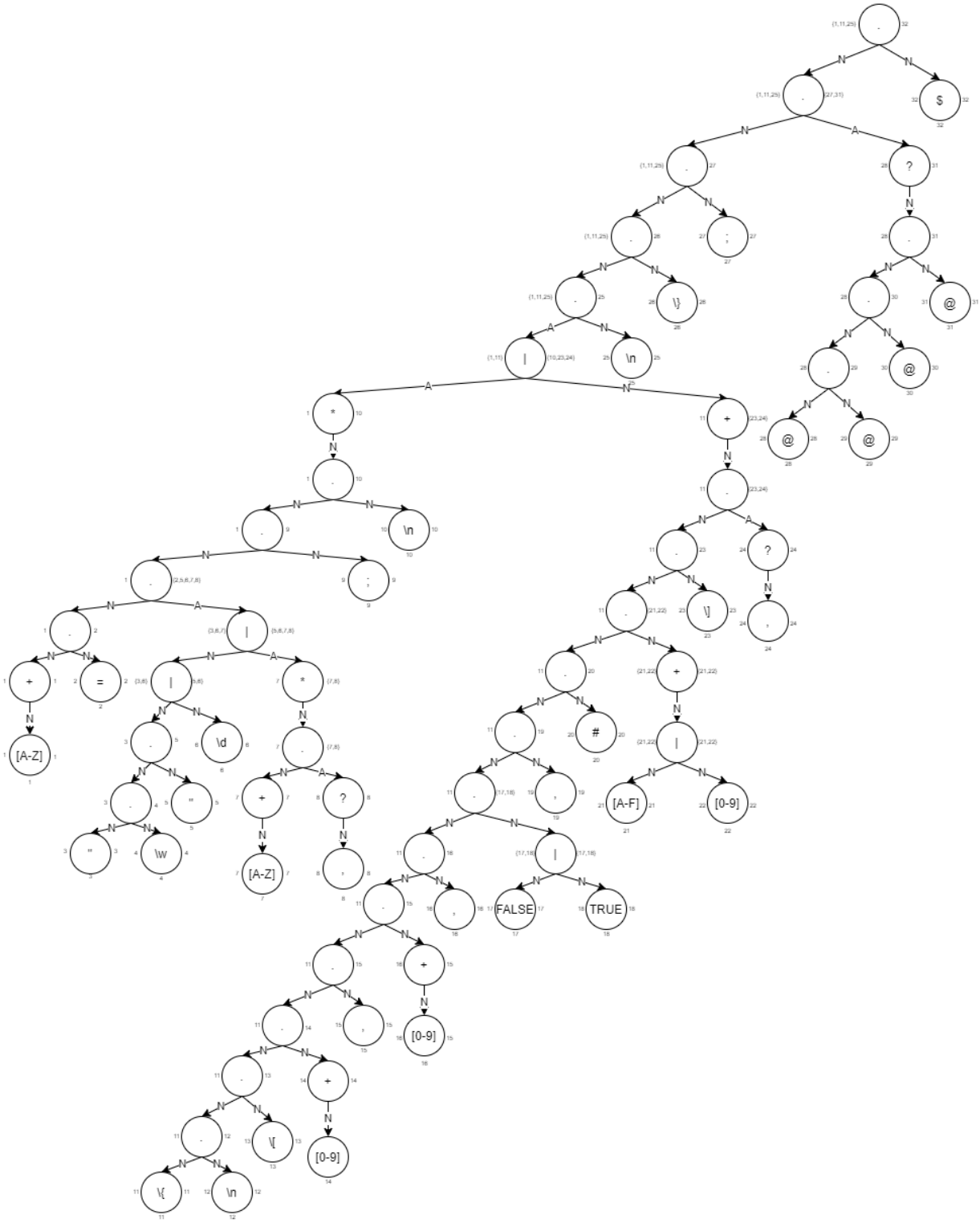
En resumen, estas son las clases y funciones en que se compone el presente programa.

## Tabla de Tokens

| No. | Token | Expresión Regular |
|---|---|---|
| 1 | Título | TITULO |
| 2 | Signo igual | = |
| 3 | Cadena | “\w” |
| 4 | Punto y coma | ; |
| 5 | Ancho | ANCHO |
| 6 | Entero | [0-9]+ |
| 7 | Alto | ALTO |
| 8 | Filas | FILAS |
| 9 | Columnas | COLUMNAS |
| 10 | Celdas | CELDAS |
| 11 | Llave de apertura | { |
| 12 | Corchete de apertura | [ |
| 13 | Coma | , |
| 14 | Boolean | TRUE\|FALSE |
| 15 | Número Hexadecimal | #[A-F]*[0-9]* |
| 16 | Corchete de cierre | ] |
| 17 | Llave de cierre | } |
| 18 | Filtro | FILTRO |
| 19 | Parámetro | MIRRORX\|MIRRORY\|DOUBLEMIRROR |
| 20 | Separadro | @@@@ |

# Árbol Binario

Expresión Regular: ^([A-Z]+=((“\w”|\d|([A-Z]+,?)*);\n)*|(\{\n\[[0-9]+,[0-9]+,(FALSE|TRUE),(#[A-F][0-9])+\],?)+\n\};)(@@@@)?$



# Autómata

Gerson Rubén Quiroa del Cid
202000166