# Manual Técnico

---

**8BIT-GIF_CREATOR**

VERSIÓN 1.0
FECHA: 27/05/2020
DEVELOPER: SOFIA GUTIERREZ
ORGANIZACIÓN: CUNOC-USAC

# 1 Descripción

El software hace uso de un total de 4 archivos de texto diferentes, cada una de estas estructuras consta de un analizador léxico y analizador sintáctico.
Los cuales sirven para traducir archivos con una estructura JSON, en datos en forma de lienzos, colores, tiempo y pintura de imágenes que el usuario puede manipular a su gusto.

# 2 Datos de Desarrollo

El Software fue desarrollado en un computador con las siguientes características:

- ★ OS: Linux Mint 19.3 ‘Tricia’ MATE
- ★ RAM: 6 GB
- ★ Procesador: Intel core i5
- ★ Lenguaje: Java 11
- ★ IDE: NetBeans 11.3
- ★ Dependencias utilizadas:
  - JAVA CUP version 11.
  - JFLEX versión 1.7.0
  - COMMONS COLLECTIONS 3.2.2

# 3 Analizadores

## 3.1 Analizadores Léxicos

Los analizadores léxicos constituyen la primera fase de un compilador, consiste en una clase que recibe como entrada el ‘código fuente’ de un programa, a través de una secuencia de caracteres y produce una salida en forma de tokens.

### 3.1.1 Lienzos

El analizador léxico de lienzos con extension **.lnz,** está constituido por los siguiente tokens en su estructura:

- ★ **REGLAS LÉXICAS**
- ➔ INT: {[0-9]}+
- ➔ ID: ({Letter} | "_")({Letter} | {Number} | "_")*
- ➔ NAME: "\"" [^*] ~"\""
- ➔ HEXCODE: ("#")({HexNumber})({HexNumber})({HexNumber})({HexNumber})({HexNumber})({HexNumber})
- ★ **PALABRAS RESERVADAS**
- ➔ LIENZOS

- ➔ nombre
- ➔ tipo
- ➔ Fondo
- ➔ Red
- ➔ Blue
- ➔ Green
- ➔ tamaño
- ➔ cuadro
- ➔ dimension_x
- ➔ dimension_Y
- ➔ HEX
- ➔ “png”
- ➔ “gif”
- ★ **SIGNOS**
- ➔ {
- ➔ }
- ➔ :
- ➔ ,

### 3.1.2 Colores

El analizador léxico de colores con extensión **.clrs,** está constituido por los siguiente tokens en su estructura:

- ★ **REGLAS LÉXICAS**
- ➔ INT: {[0-9]}+
- ➔ ID: ({Letter} | "_")({Letter} | {Number} | "_")*
- ➔ HEXCODE: ("#")({HexNumber})({HexNumber})({HexNumber})({HexNumber})({HexNumber})({HexNumber})
- ★ **PALABRAS RESERVADAS**
- ➔ COLORES
- ➔ Red
- ➔ Blue
- ➔ Green
- ➔ HEX
- ★ **SIGNOS**
- ➔ {
- ➔ }
- ➔ :
- ➔ ,

### 3.1.3 Tiempo

El analizador léxico de tiempo con extensión **.tmp,** está constituido por los siguiente tokens en su estructura:

★ **REGLAS LÉXICAS**
➔ INT: {[0-9]}+
➔ IDENTIFICADOR: ({Letter} | "_")({Letter} | {Number} | "_")*
★ **PALABRAS RESERVADAS**
➔ TIEMPOS
➔ inicio
➔ fin
➔ imagenes
➔ id
★ **SIGNOS**
➔ {
➔ }
➔ :
➔ ,
➔ “
➔ [
➔ ]

### 3.1.4 Pintar

El analizador léxico de pintar con extensión **.pnt,** está constituido por los siguiente tokens en su estructura:

★ **REGLAS LÉXICAS**
➔ INT: {[0-9]}+
➔ ID: ({Letter} | "_")({Letter} | {Number} | "_")*
➔ STRING: "\"" [^*] ~"\""
➔ RANK: ({Number})+("..")({Number})+
➔ TRADITIONALCOMMENTARY: "/*" [^*] ~"*/"
➔ BASICCOMMENTARY: "//" {InputCharacter}* {LineTerminator}?
➔ Commentary: {TraditionalCommentary} | {BasicCommentary}
★ **PALABRAS RESERVADAS**
➔ VARS
➔ int
➔ String
➔ boolean
➔ INSTRUCCIONES
➔ PINTAR
➔ true
➔ false
➔ AND
➔ OR
➔ while
➔ if
➔ else

★ **SIGNOS**

➔ {
➔ }
➔ [
➔ ]
➔ (
➔ )
➔ .
➔ :
➔ .
➔ =
➔ ==
➔ <
➔ >
➔ <=
➔ =>
➔ <>
➔ +
➔ -
➔ *
➔ /

## 3.2 Analizadores Sintácticos

Parte media del compilador, transforma la entrada de un árbol de derivación. El análisis sintáctico convierte los tokens recibidos por el analizador léxico en otras estructuras, entre las cuales la más común son los árboles, ya que son útiles para el análisis posterior y capturar la jerarquía implícita en la entrada.

### 3.2.1 Gramática Lienzo

**G** = {N,T,P,S}

**N** = { starting, canvasStructure, specificCanvas, name, type, extension, background, backgroundType, rgbBackground, hexBackground, redPermutations, greenPermutations, bluePermutations, red, green, blue, square, x, y, size, sizeStructure, squarePermutations, xPermutations, yPermutations, structure, namePermutations, sizePermutations, typePermutations, backgroundPermutations }

**T** = { CURLYBRACKETO, CURLYBRACKETC, LIENZOS, ID, NOMBRE, NAME, TIPO, PNG, GIF, FONDO, RED, BLUE, GREEN, SIZE, COLON, COMMA, CUADRO, DIMENSIONX, DIMENSIONY, HEX, HEXCODE, INT }

**S** = starting.

**P =**
starting -> CURLYBRACKETO canvasStructure CURLYBRACKETC

canvasStructure -> LIENZOS COLON CURLYBRACKETO specificCanvas CURLYBRACKETC

specificCanvas -> ID:id COLON CURLYBRACKETO structure:canvas CURLYBRACKETC COMMA specificCanvas
| ID:id COLON CURLYBRACKETO structure:canvas CURLYBRACKETC

structure -> name:name COMMA namePermutations:canvas
| type:type COMMA typePermutations:canvas
| size:size COMMA sizePermutations:canvas
| background:color COMMA backgroundPermutations:canvas

name -> NOMBRE COLON NAME:name

type -> TIPO COLON extension:extension

extension -> PNG:png
| GIF:gif

background -> FONDO COLON CURLYBRACKETO backgroundType:color CURLYBRACKETC

backgroundType -> rgbBackground:color
| hexBackground:color

rgbBackground -> red:red COMMA redPermutations:color
| green:green COMMA greenPermutations:color
| blue:blue COMMA bluePermutations:color

red -> RED COLON INT:integer
green -> GREEN COLON INT:integer
blue -> BLUE COLON INT:integer

redPermutations -> green:green COMMA blue:blue
| blue:blue COMMA green:green

greenPermutations -> red:red COMMA blue:blue
| blue:blue COMMA red:red

```
bluePermutations -> green:green COMMA red:red
                  | red:red COMMA green:green


hexBackground -> HEX COLON HEXCODE:hexcode {: RESULT = Color.decode(hexcode) :}

size -> SIZE COLON CURLYBRACKETO sizeStructure:square CURLYBRACKETC
sizeStructure -> square:size COMMA squarePermutations:square
                  | x:x COMMA xPermutations:square
                  | y:y COMMA yPermutations:square


square -> CUADRO COLON INT:size

x -> DIMENSIONX COLON INT:x


y -> DIMENSIONY COLON INT:y
squarePermutations -> x:x COMMA y:y
                  | y:y COMMA x:x

xPermutations -> square:square COMMA y:y
                  | y:y COMMA square:square

yPermutations -> square:square COMMA x:x
                  | x:x COMMA square:square

namePermutations -> background:color COMMA size:size COMMA type:type
                  | background:color COMMA type:type COMMA size:size
                  | size:size COMMA type:type COMMA background:color
                  | size:size COMMA background:color COMMA type:type
                  | type:type COMMA size:size COMMA background:color
                  | type:type COMMA background:color COMMA size:size

sizePermutations -> background:color COMMA name:name COMMA type:type
                  | background:color COMMA type:type COMMA name:name
                  | name:name COMMA background:color COMMA type:type
                  | name:name COMMA type:type COMMA background:color
                  | type:type COMMA background:color COMMA name:name
                  | type:type COMMA name:name COMMA background:color

backgroundPermutations -> name:name COMMA type:type COMMA size:size
                  | name:name COMMA size:size COMMA type:type
                  | type:type COMMA name:name COMMA size:size
                  | type:type COMMA size:size COMMA name:name
```

| size:size COMMA name:name COMMA type:type
| size:size COMMA type:type COMMA name:name

typePermutations -> name:name COMMA background:color COMMA size:size
| name:name COMMA size:size COMMA background:color
| background:color COMMA name:name COMMA size:size
| background:color COMMA size:size COMMA name:name
| size:size COMMA background:color COMMA name:name
| size:size COMMA name:name COMMA background:color

### 3.1.2 Gramática Colores

**G** = {N,T,P,S}

**N** = { starting, colorsStructure, canvasColors, anotherColor, endingCanvas, specificColor hexCode, redPermutations, greenPermutations, bluePermutations, rgbCode, tone, red, green, blue }

**T** = { CURLYBRACKETO, CURLYBRACKETC, COLORES, ID, RED, BLUE, GREEN, COLON, COMMA, HEX, HEXCODE INT }

**S** = starting.
**P** =
starting -> CURLYBRACKETO colorsStructure CURLYBRACKETC

colorsStructure -> COLORES COLON CURLYBRACKETO canvasColors CURLYBRACKETC

canvasColors -> ID:id COLON CURLYBRACKETO specificColor endingCanvas

specificColor -> ID:id COLON CURLYBRACKETO tone:color anotherColor

tone -> rgbCode:color
| hexCode:color

rgbCode -> red:red COMMA redPermutations:color
| green:green COMMA greenPermutations:color
| blue:blue COMMA bluePermutations:color

red -> RED COLON INT:integer
green -> GREEN COLON INT:integer
blue -> BLUE COLON INT:integer

redPermutations -> green:green COMMA blue:blue
| blue:blue COMMA green:green

greenPermutations -> red:red COMMA blue:blue
| blue:blue COMMA red:red

bluePermutations -> green:green COMMA red:red
| red:red COMMA green:green

hexCode -> HEX COLON HEXCODE:hexcode

anotherColor -> CURLYBRACKETC COMMA specificColor
| CURLYBRACKETC

endingCanvas -> CURLYBRACKETC COMMA canvasColors
| CURLYBRACKETC

### 3.1.3 Gramática Tiempo

**G** = {N,T,P,S}

**N** = { starting, timesStructure, canvasTimes, anotherImage, imagesStructure, endingCanvas, idStructure, starter, end, duration, imagesPermutations, images, structurePermutations }

**T** = { CURLYBRACKETO, CURLYBRACKETC, TIEMPOS, ID, INICIO, FIN, IMAGENES, COLON, COMMA, DURACION, QUOTE, BRACKETC, BRACKETO, IDENTIFICADOR, INT }

**S** = starting.
**P** =
starting -> CURLYBRACKETO timesStructure CURLYBRACKETC
timesStructure -> TIEMPOS COLON CURLYBRACKETO canvasTimes CURLYBRACKETC
canvasTimes -> IDENTIFICADOR COLON CURLYBRACKETO structurePermutations endingCanvas
structurePermutations -> starter COMMA end COMMA images
| starter:start COMMA images COMMA end:end
| end:end COMMA starter:start COMMA images
| end:end COMMA images COMMA starter:start
| images COMMA starter:start COMMA end:end
| images COMMA end:end COMMA starter:start

starter -> INICIO COLON QUOTE IDENTIFICADOR:id QUOTE

end -> FIN COLON QUOTE IDENTIFICADOR:id QUOTE

```
images -> IMAGENES COLON BRACKETO imagesStructure BRACKETC

imagesStructure -> CURLYBRACKETO imagesPermutations:image anotherImage

anotherImage -> CURLYBRACKETC COMMA imagesStructure
                | CURLYBRACKETC

imagesPermutations -> idStructure:id COMMA duration:duration
                | duration:duration COMMA idStructure:id

idStructure -> ID COLON QUOTE IDENTIFICADOR:id QUOTE
duration -> DURACION COLON INT:integer
endingCanvas -> CURLYBRACKETC COMMA canvasTimes
                | CURLYBRACKETC
```

## 3.3 Analizadores Semánticos

Un analizador semántico es el encargado de verificar la compatibilidad entre un operador y sus operandos, que el flujo sea adecuado, que no haya duplicidad entre nombres, etc

Hay dos tipos, uno estático, en tiempo de compilación y otro dinámico en tiempo de ejecución.

El análisis semántico dentro del código se realizó mediante un árbol de expresiones, en el caso del archivo pintura, mientras que en el resto de archivos se realizó mediante una tabla de símbolos.