UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
LENGUAJES FORMALES Y DE PROGRAMACIÓN

Primer Semestre 2016

Sección	Catedrática	Tutor académico
A-	Inga. Damaris Campos de López	Eiji de Paz
B-	Inga. Zulma Aguirre	Luis Fernando Lara Lemus

Enunciado de Proyecto No. 1

Objetivos:

- Crear un analizador léxico que pueda obtener información a partir de un archivo de entrada.
- Practicar el lenguaje de Programación Visual Basic y conocer algunas de sus funciones con el manejo de imágenes.
- Implementar, por medio del método del árbol, un autómata finito determinístico para la resolución del problema.

Descripción:

El proyecto consiste en realizar un creador de imágenes en mapa de bits, en la actualidad aún es posible observar imágenes con profundidad de color de 8 bits, en juegos o películas, el mapa de bits consiste en una matriz de píxeles que contiene puntos de color en cada una de sus posiciones.

Por lo tanto, se requiere que por medio de un lenguaje de alto nivel se puedan crear, editar y guardar este tipo de imágenes, la aplicación será desarrollada en el lenguaje de programación Visual Basic.Net.

La aplicación contará con un editor de texto, desde el cual se podrán crear o bien editar las imágenes por medio de un lenguaje posteriormente explicado.

También deberá generar la información del análisis léxico en una página HTML al igual que los errores léxicos, si existieran.

Definición del Lenguaje:

El lenguaje que se analizará cuenta con los estándares de un archivo JSON, que es, un tipo de archivo utilizado generalmente para el intercambio de información entre lenguajes.

La estructura para el archivo de entrada es el siguiente:

- **Objeto:** se especifica con la palabra reservada 'imagen', con esto le estamos diciendo a nuestra aplicación que queremos crear una imagen.
- **Atributos:**
  - **Nombre**: nombre con el cual se guardará nuestra imagen.
  - **Alto**: representa el número de filas que contendrá el mapa de bit´s.
  - **Ancho**: representa el número de columnas que contendrá el mapa de bit´s.
  - **Formato:** Es el tipo de formato de salida que tendrá nuestra imagen, puede ser 'jpg', 'png' o 'bmp'
  - **Colores**: contiene un arreglo especificando el color de cada pixel en nuestro mapa de bit.
    - **X:** coordenada X (columna) en nuestra matriz de colores (mapa de bits)
    - **Y:** coordenada Y (fila) en nuestra matriz de colores (mapa de bits)
    - **Color:** representa el color que se pintará en nuestra matriz de colores (mapa de bits), puede ser: rojo, amarillo, blanco, negro, café, anaranjado, azul, verde.
- **Comentarios:** Es posible incluir comentarios a nuestro archivo, para documentarlo, solo se permiten comentarios de 1 sola línea e inician con '//'
- **NOTA**: los atributos para la imagen, pueden estar en cualquier orden.

- Para los atributos ‘alto’, ‘ancho’, ‘x’, ‘y’, que reconocen números como valor, también será posible ingresar alguna ecuación para indicar el valor.

Precedencia	Operación	Operador
1	Suma	+
1	Resta	-
2	Multiplicación	*
2	División	/
3	Agrupación	()

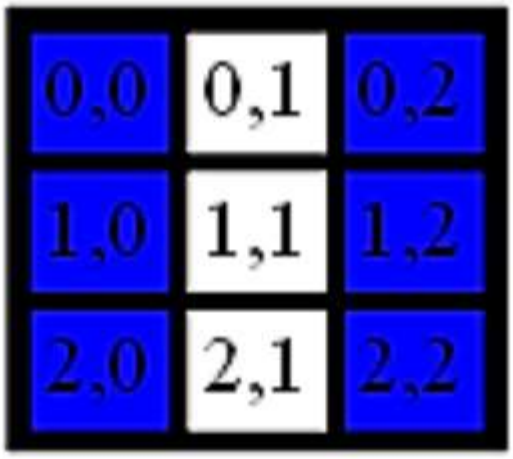
- **NOTA**: La matriz de colores NO siempre será cuadrada, puede ser cualquier número comprendido del 1 al 10.
- Ejemplos de ecuaciones validas:

Ecuación	Resultado
2*5+1	11
(2+4)*2	12
10/5*3	6

A continuación, se muestra un ejemplo del archivo de entrada con su respectiva imagen de salida:

```
{
  //inicio de la imagen
  "imagen": {
    "nombre": "bandera",
    "alto": 2+1,
    "ancho": (3*1)+0,
    "formato": "jpg",
    "colores":[ //colores de relleno de matriz
      {"y":0, "x":0, "color":"azul" },
      {"y":0, "x":1, "color":"blanco"},
      {"y":0, "x":2, "color":"azul"},
      {"y":1, "x":0, "color":"azul"},
      {"y":1, "x":1, "color":"blanco"},
      {"y":1, "x":2, "color":"azul" },
      {"y":2, "x":0, "color":"azul"},
      {"y":2, "x":1, "color":"blanco"},
      {"y":2, "x":2, "color":"Azul"},
      }
      ]
  }
}
```

Imagen:



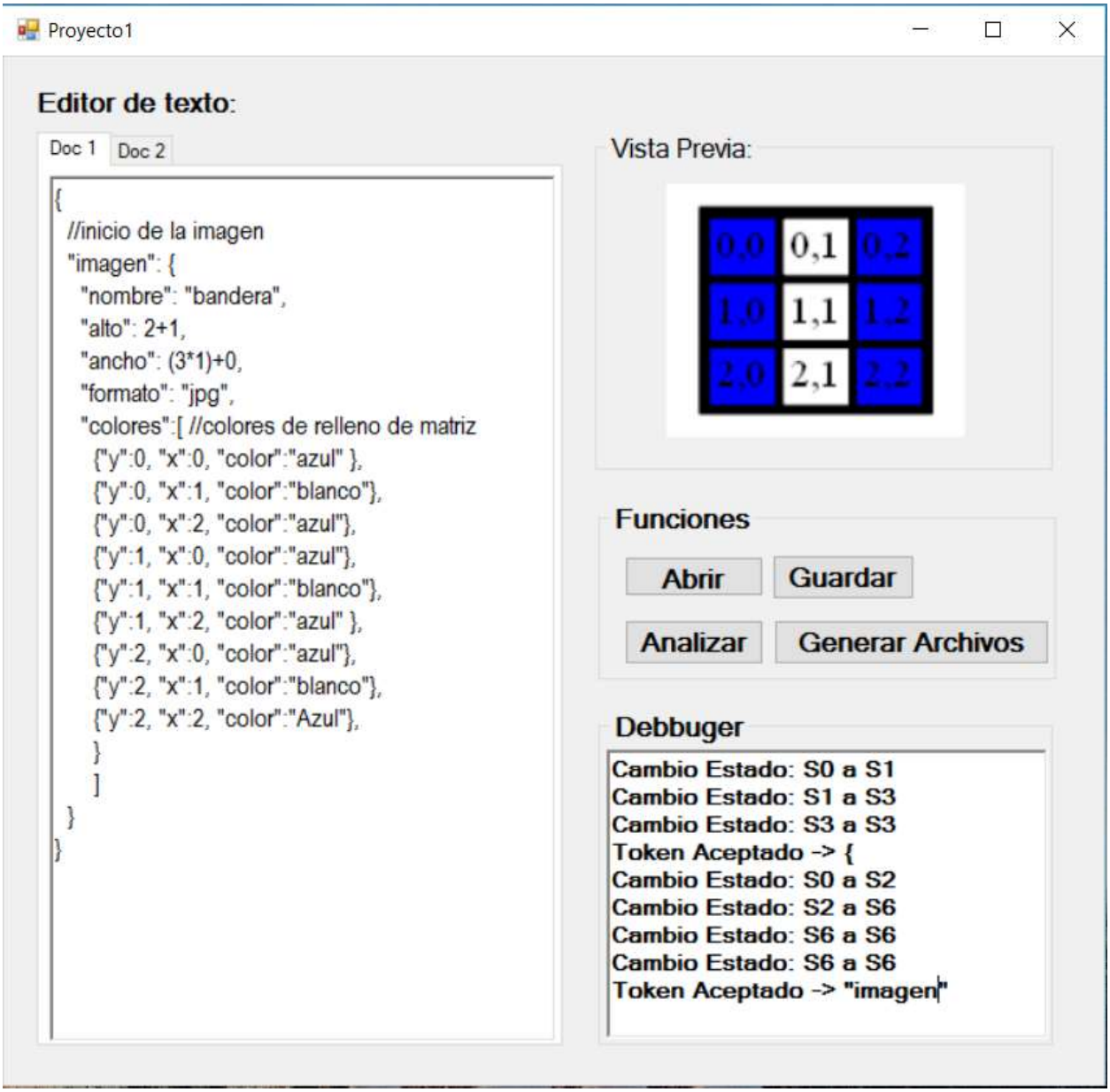
Interfaz gráfica:

La interfaz gráfica se compone de los siguientes elementos:

- Editor de Texto
  - Campo de texto donde se podrá ingresar los archivos de entrada o bien escribirlos.
  - Deberá ser posible escribir distintos archivos en distintas pestañas.
  - NOTA: el editor de texto, siempre y cuando no existan errores, deberá pintar cada uno de los lexemas según la siguiente tabla:

Tipo	Color
Identificadores	Verde
Números	Azul
Cadenas	Amarillo
Palabra reservada	Morado
Comentarios	Rojo
Otros….	Indicar el color en el manual del usuario

- Vista Previa
  - Aquí se mostrará la imagen después de haber analizado el archivo y haber presionado el botón 'Analizar'.
- Funciones
  - **Abrir**: abre un archivo de texto que contendrá el código de alto nivel.
  - **Guardar**: guarda en un archivo de texto con el contenido del editor (pestaña actual); deberá poderse especificar la ruta.
  - **Analizar**: inicia el análisis léxico y posteriormente genera la imagen y la muestra en 'vista previa'
  - **Generar Archivos**: genera la tabla de tokens o bien los archivos de errores en las páginas HTML correspondientes. Y se genera un archivo con la bitácora (función debugger).
- Debugger
  - En este módulo se deberá implementar una bitácora del recorrido del autómata, mostrando los cambios de estados y los tokens que se acepten o errores encontrados.
- Menú
  - Ayuda
    - **Acerca de**: deberá mostrar los datos del estudiante.
    - **Manual**: desplegará el pdf con el manual de usuario del estudiante.
  - Archivo:
    - **Nuevo**: borra todo lo que esté en la pestaña actual del editor de texto para empezar un Nuevo documento.
    - **Salir**: se sale de la aplicación.



Archivos de Salida:

El archivo de Tokens se debe mostrar en una página HTML, la información en una tabla como se muestra a continuación:

#	Fila	Columna	Lexema	Id Token	Token
1	09	25	azul	11	ColorPixel
2	02	02	//Inicio de la imagen	12	Comentario
3	07	15	jpg	13	FormatoImagen

El archivo de Errores, si existieran, se debe mostrar en una página HTML con la información en una tabla como se muestra a continuación:

#	Fila	Columna	Carácter	Descripción
1	05	10	|	Desconocido
2	08	30	¬	Desconocido
3	10	05	~	Desconocido

Entregables que se deben incluir en el CD:

- Manual de Usuario
- Manual Técnico, debe incluir la definición de los tokens (IdToken, Token, patrón –Exp.Reg.-) y el DFA (por el método del árbol a partir de las Expresiones Regulares de los tokens definidos) que se use para el analizador léxico.
- Código Fuente
- Ejecutable de la Aplicación.

Documentación a entregar de forma física el día de la calificación:

- Hoja de calificación (Original y una copia)

Notas importantes:

- La práctica se debe desarrollar de forma individual.
- Esta práctica se deberá desarrollar utilizando Visual Basic .Net con Visual Studio 2013.
- Para la creación de la imagen (mapa de bits) solamente es permitido utilizar el programa "graphviz", para su descarga y acceso a la documentación:

  www.graphviz.org
- El proceso de obtener tokens, se debe hacer a través de la implementación del autómata finito determinista obtenido por medio del método del árbol y desarrollado por el propio estudiante.
- No se puede agregar o quitar algún símbolo en el archivo de entrada. El proyecto deberá funcionar con los archivos de prueba que se disponga para la calificación, sin modificación.
- La calificación de la práctica será personal y durará como máximo 30 minutos, en un horario que posteriormente será establecido. Se debe tomar en cuenta que durante la calificación no podrán estar terceras personas alrededor, de lo contrario no se calificará la práctica.
- No se dará prórroga para la entrega de la práctica.
- **Copia parcial o total tendrá una nota de 0 puntos, y se notificará a la Escuela de Sistemas para que se apliquen las sanciones correspondientes.**
- **En el caso de no cumplir con alguna de las indicaciones antes mencionadas, NO se calificará el proyecto; por lo cual, se tendrá una nota de cero puntos.**

Fecha de entrega: 10 de Marzo de 2016