

**Tabla de Contenidos**

Descripción del Problema.....2

Diseño del programa.....3

Librerías externas usadas.....4

Análisis de resultados.....5

Manual de Usuario.....6

Conclusión Personal.....8

## Descripción del Problema

Se deberá construir un analizador léxico para XHTML, por lo cual se debe investigar la especificación estándar de XHTML. También se deberá escribir la especificación en Flex, definir la lista de tokens que van a ser necesarios para crear un analizador léxico para XHTML, generar las expresiones regulares y las acciones asociadas a estas para el procesamiento de los tokens.

## Diseño del programa

Se decidió utilizar Flex por que se encontró más información y ejemplos de este, además fue el generador de analizadores léxicos que se vio en clase.

Con respecto a las expresiones regulares se decidió hacer una expresión regular con todos los “elementos” al igual con los “atributos” de XHTML, y la acción asociado retornaba ya sea T\_Element o T\_Atributte y el elemento o atributo se almacenaba en la variable yylval.

Lista de tokens:

- T\_OpenSign: Token del símbolo “<”.
- T\_CloseSign: Token del símbolo “>”.
- T\_Elements: Token de todos los elementos de XHTML.
- T\_Atributtes: Token de todos los atributos de XHTML.
- T\_Values: Token del valor de un atributo.
- T\_Text: Token del texto.
- T\_Minus: Token del símbolo “-”.
- T\_Equal: Token de asignación “=”.
- T\_Colon : Token del símbolo “:”.
- T\_Slash: Token del símbolo “/”.
- T\_QuestionMark: Token del símbolo “?”.
- T\_ExclamationMark: Token del símbolo “!”.
- T\_PublicIdentifier: Token de la palabra reservada “PUBLIC”.
- T\_DOCTYPE: Token de la palabra reservada “DOCTYPE”.

## Librerías externas usadas

- `scanner.h`: en este archivo de cabecera es donde se definen los diferentes tokens que son retornados en las distintas reglas definidas en el scanner.
- `stdio.h`: en este archivo de cabecera es donde se definen los macros, constantes, declaraciones de funciones y la definición de tipos usados por varias operaciones estándar de entrada y salida,

## **Análisis de resultados**

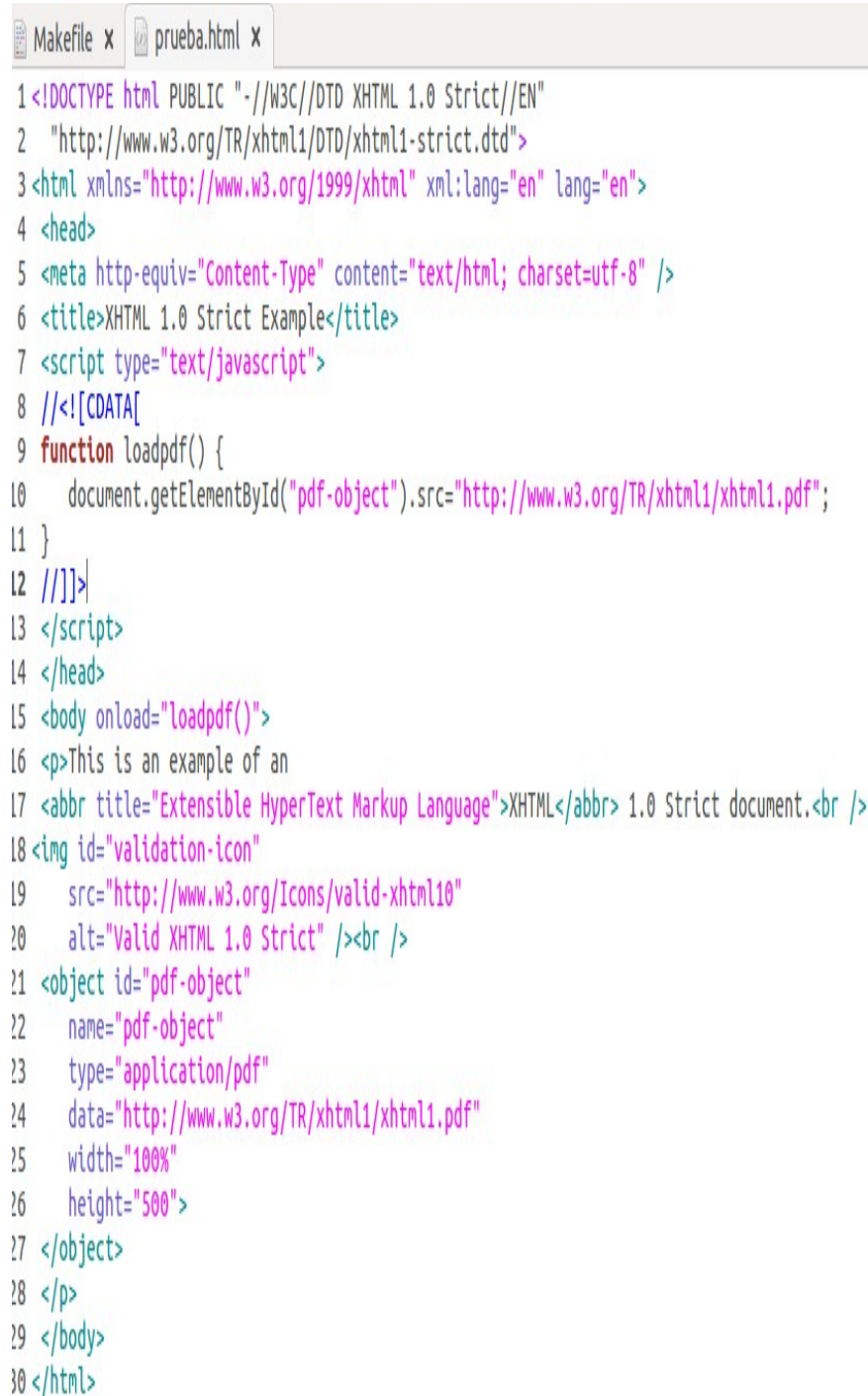
A continuación se lista los objetivos del proyecto y la descripción del nivel de alcance que se logró en cada uno:

- Verificar las palabras reservadas: Este objetivo se logró, ya que el programa verifica todas las palabras reservadas y las almacena en una variable.
- Verificar comentarios: El programa procesa los comentarios de tal manera que si encuentra un comentario sin cerrar o algún error léxico, genera un error.
- Verificar texto: Todo el texto que no presente errores léxicos es procesado y almacenado en una variable.
- Reporte errores: Los errores que se presenten al analizar el archivo o el texto XHTML se reporta al stderr.

# Manual de Usuario

A continuación se muestran los pasos a seguir para la ejecución del programa y la interpretación de los resultados:

1. Se puede utilizar cualquier archivo como entrada, pero para poder ver el funcionamiento del analizador léxico de XHTML se utiliza un archivo XHTML como este:



```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6 <title>XHTML 1.0 Strict Example</title>
7 <script type="text/javascript">
8 //
9 function loadpdf() {
10     document.getElementById("pdf-object").src="http://www.w3.org/TR/xhtml1/xhtml1.pdf";
11 }
12 //]]&gt;
13 &lt;/script&gt;
14 &lt;/head&gt;
15 &lt;body onload="loadpdf()"&gt;
16 &lt;p&gt;This is an example of an
17 &lt;abbr title="Extensible HyperText Markup Language"&gt;XHTML&lt;/abbr&gt; 1.0 Strict document.&lt;br /&gt;
18 &lt;img id="validation-icon"
19     src="http://www.w3.org/Icons/valid-xhtml10"
20     alt="Valid XHTML 1.0 Strict" /&gt;&lt;br /&gt;
21 &lt;object id="pdf-object"
22     name="pdf-object"
23     type="application/pdf"
24     data="http://www.w3.org/TR/xhtml1/xhtml1.pdf"
25     width="100%"
26     height="500"&gt;
27 &lt;/object&gt;
28 &lt;/p&gt;
29 &lt;/body&gt;
30 &lt;/html&gt;</pre></div>
```

2. Compilación y ejecución: Mediante el archivo makefile que se encarga de realizar llamadas a los compiladores, operaciones de limpieza y demás, para generar un ejecutable llamado scxt al cual se le indica un archivo que servirá como entrada mediante el operador de redirección < y otro como salida utilizando el operador para redirección >, como se observa en el ejemplo respectivamente:

```
Jack@jack-CX610: ~/IC-5701-TP1-JeanCarloCervantes
jack@jack-CX610:~$ cd IC-5701-TP1-JeanCarloCervantes
jack@jack-CX610:~/IC-5701-TP1-JeanCarloCervantes$ make
gcc -o scxt lex.yy.o main.o -lc -lm -ll
jack@jack-CX610:~/IC-5701-TP1-JeanCarloCervantes$ ./scxt < prueba.html > tokenresultante.txt
jack@jack-CX610:~/IC-5701-TP1-JeanCarloCervantes$
```

3. Interpretación de la salida: Seguidamente se obtiene una salida como el archivo que se muestra a continuación, en el cual se encuentra la clasificación de los diferentes tokens y su valor respectivo de ser necesario, ignorando espacios en blanco y saltos de línea.

```
tokenresultante.txt x
1 (name= T_OpenSign)
2 (name= T_ExclamationMark)
3 (name= T_DOCTYPE)
4 (name= T_Elements , value = html)
5 (name= T_PublicIdentifier)
6 (name= T_Values , value = "-//W3C//DTD XHTML 1.0 Strict//EN")
7 (name= T_Values , value = "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd")
8 (name= T_CloseSign)
9 (name= T_OpenSign)
10 (name= T_Elements , value = html)
11 (name= T_Atributtes , value = xmlns)
12 (name= T_Equal)
13 (name= T_Values , value = "http://www.w3.org/1999/xhtml")
14 (name= T_Atributtes , value = xml)
15 (name= T_Colon)
16 (name= T_Atributtes , value = lang)
17 (name= T_Equal)
18 (name= T_Values , value = "en")
19 (name= T_Atributtes , value = lang)
20 (name= T_Equal)
21 (name= T_Values , value = "en")
22 (name= T_CloseSign)
23 (name= T_OpenSign)
24 (name= T_Elements , value = head)
25 (name= T_CloseSign)
26 (name= T_OpenSign)
27 (name= T_Elements , value = meta)
28 (name= T_Atributtes , value = http-equiv)
29 (name= T_Equal)
30 (name= T_Values , value = "Content-Type")
31 (name= T_Atributtes , value = content)
32 (name= T_Equal)
33 (name= T_Values , value = "text/html; charset=utf-8")
34 (name= T_Slash)
35 (name= T_CloseSign)
36 (name= T_OpenSign)
```

## Conclusión Personal

1. El desarrollo de un scanner es simple si se utilizan expresiones regulares, tomando en cuenta que cualquier error mínimo supone un problema en los siguientes analizadores.
2. El generador de analizadores léxicos Flex es muy fácil de aprender y de utilizar, esto resulta muy útil para el objetivo de esta tarea programada, ya que no se desperdicia tiempo en mucha investigación.
3. El uso de headers hace la programación mas ordenada y útil, para detectar problemas de forma mas sencilla y rápida.