**Ingeniería en sistemas de Información**

**Sintaxis y semántica de los lenguajes**

**TRABAJO PRACTICO INTEGRADOR:**

**Universidad Tecnológica Nacional FRRE**

**Curso académico 2020**

**Profesores:**

* Ing. Gabriela Tomaselli.
* Ing. Vigil, Rodrigo.
* Ing. Tortosa, Nicolas G.
* Ing. Torre, Juliana.

**Alumno:**

* Mass, Matias.

**Comisión:** k2.1

**Grupo:** 21.15

Contenido

[**1. Introducción:** 3](#_Toc44179048)

[**2. Diseño de gramática y tipos de etiquetas:** 3](#_Toc44179049)

[**3. Analizador léxico:** 7](#_Toc44179050)

[**4. Analizador Sintáctico:** 8](#_Toc44179051)

[**5. Flex y Bison:** 8](#_Toc44179052)

[**6. Funciones y librerías utilizadas:** 10](#_Toc44179053)

[**7. Consideraciones:** 11](#_Toc44179054)

[**8. Ejecución del analizador léxico y sintáctico:** 11](#_Toc44179055)

[**9. Bibliografía:** 12](#_Toc44179056)

# **1. Introducción:**

El trabajo practico integrador de la materia: sintaxis y semántica de los lenguajes, consiste en poder realizar un intérprete del lenguaje HTML mediante los generadores de lexer y parser.

HTML (HyperText Markup Language) es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet y su fecha de lanzamiento inicial fue en el año 1992.

El trabajo practico integrador se divide en 3 etapas:

1. La primera etapa consta de crear una gramática libre de contexto donde contenga las etiquetas/tags del lenguaje de marcado HTML.
2. La segunda etapa consta de utilizar alguna herramienta para el análisis lexicográfico para las expresiones regulares, tokens, patrones y lexemas que componen al lenguaje.
3. La tercera etapa consta de utilizar otra herramienta para el análisis sintáctico y una vez hecho todas las etapas anteriores, se pide crear un archivo del tipo .exe para ejecutar las instrucciones para analizar léxica y semánticamente archivos del tipo .HTML o .txt.

# **2. Diseño de gramática y tipos de etiquetas:**

Para la primera etapa del trabajo practico integrador se implementó lo aprendido en clases para realizar el diseño de gramática del lenguaje de programación HTML.

La gramática que se diseño es libre de contexto ya que los lenguajes de programación son generados por estas gramáticas. Se tuvo en cuenta los terminales y no terminales del lenguaje, como así también la recursividad de los no terminales que permite generar infinitas líneas de instrucciones que el usuario quiera. Se evite que hubiera producciones que generen ambigüedad.

|  |  |  |
| --- | --- | --- |
| Σ |  | <!doctype html>INICIO |
| INICIO |  | <html lang=”idioma”>HEADBODY</html> |
| HEAD |  | <head></head> | <head>META</head> |
|  | <head>METATITLE</head> | <head>TITLEMETA</head> |
| TITLE |  | <title></title> | <title>texto</title> |
| META |  | <meta charset=”utf-8”> | <meta charset=”utf-8”>METAS |
| METAS |  | <meta ATRIMT”> | <meta ATRIMT>METAS |
| ATRIMT |  | name=”texto” content=”texto” |
| BODY |  | <body></body> | <body>ETIQ</body> |
| ETIQ |  | texto | textoETIQ |
|  | A | AETIQ |
|  | H | HETIQ |
|  | P | PETIQ |
|  | BR | BRETIQ |
|  | EM | EMETIQ |
|  | HR | HRETIQ |
|  | OL | OLETIQ |
|  | UL | ULETIQ |
|  | DIV | DIVETIQ |
|  | IMG | IMGETIQ |
|  | MARK | MARKETIQ |
|  | TABLE | TABLEETIQ |
|  | STRONG | STRONGETIQ |
|  | SECTION | SECTIONETIQ |
| A |  | <a ATA></a> |
|  | <a ATA>ETIQ</a> |
| H |  | <h1>texto</h1> | <h1 ATG>texto</h1> |
|  | <h2>texto</h2> | <h2 ATG>texto</h2> |
|  | <h3>texto</h3> | <h3 ATG>texto</h3> |
|  | <h4>texto</h4> | <h4 ATG>texto</h4> |
|  | <h5>texto</h5> | <h5 ATG>texto</h5> |
|  | <h6>texto</h6> | <h6 ATG>texto</h6> |
| P |  | <p></p> | <p ATG></p> | <p>ETIQ</p> | <p ATG>ETIQ</p> |
| BR |  | <br> | <br ATG> |
| EM |  | <em></em> | <em ATG></em> | <em>ETIQ</em> | <em ATG>ETIQ</em> |
| HR |  | <hr> | <hr ATG> |
| DIV |  | <div></div> | <div ATG></div> | <div>ETIQ</div> | <div ATG>ETIQ</div> |
| IMG |  | <img ATIMG> | <img ATG ATIMG> |
| OL |  | <ol>LISTA</ol> | <ol ATOL>LISTA</ol> |
| UL |  | <ul>LISTA</ul> | <ul ATG>LISTA</ul> |
| LISTA |  | LI | LILISTA |
| LI |  | <li></li> | <li ATLI></li> | <li>ETIQ</li> | <li ATLI>ETIQ</li> |
| MARK |  | <mark></mark> | <mark ATG></mark> | <mark>ETIQ</mark> | <mark ATG>ETIQ</mark> |
| TABLE |  | <table>TCUERPO</table> | <table ATABLE>TCUERPO</table> |
|  | <table>CAPTIONTCUERPO</table> |
|  | <table ATABLE>CAPTIONTCUERPO</table> |
| CAPTION |  | <caption>texto</caption> | <caption ATG>texto</caption> |
| TCUERPO |  | B | F | HD |
|  | BHD | HDB |
|  | BF | FB |
|  | FHD | HDF |
|  | BHDF |BFHD |
|  | HDBF | HDFB |
|  | FHDB | FBHD |
|  | TR |
| HD |  | <thead>TR</thead> | <thead ATG>TR</thead> |
| B |  | <tbody>TR</tbody> | <tbody ATG>TR</tbody> |
| F |  | <tfoot>TR</tfoot> | <tfoot ATG>TR</tfoot> |
| TR |  | <tr>TS</tr> | <tr>TS</tr>TR |
|  | <tr ATG>TS</tr> | <tr ATG>TS</tr>TR |
| TS |  | TH | THTS | TD | TDTS |
| TH |  | <th></th> | <th ATG></th> | <th>ETIQ</th> | <th ATG>ETIQ</th> |
| TD |  | <td></td> | <td ATG></td> | <td>ETIQ</td> | <td ATG>ETIQ</td> |
| STRONG |  | <strong></strong> | <strong ATG></strong> |
|  | <strong>ETIQ</strong> | <strong ATG>ETIQ</strong> |
| SECTION |  | <section></section> | <section ATG></section> |
|  | <section>ETIQ</section> | <section ATG>ETIQ</section> |
| ATG |  | class=”texto” | id=”texto” | class=”texto” id=”texto” |
| ATA |  | href=”url” | href=”url” target=”targs” | ATG href=”url” target=”targs” |
| ATLI |  | value=”tipos” | ATG | ATG value=”numeros” |
| ATOL |  | type=”tipos” | ATG | ATG type=”tipos” |
| ATABLA |  | border=”texto” | ATG | ATG border=”texto” |
| ATIMG |  | src=”url” | src=”url” height=”numeros” | src=”url” width=”numeros” |
|  | src=”url” width=”numeros” height=”numeros” | src=”url” alt=”texto" |
|  | alt=”texto" src=”url” | alt=”texto” src=”url” width=”numeros” |
|  | alt=”texto" src=”url” width=”numeros” height=”numeros” |
|  | alt=”texto" src=”url” height=”numeros” |

**Etiquetas utilizadas:**

Debido a que el lenguaje de marcado HTML tiene muchas etiquetas o tags que se pueden utilizar a la hora de crear una página web, se limitó solo a la utilización de las siguientes etiquetas:

* <!DOCTYPE html>: define que el documento está bajo el estándar de HTML 5.
* <html>: representa la raíz de un documento HTML o XHTML. Todos los demás elementos deben ser descendientes de este elemento.
* <a>: representa un hiperenlace, enlazado a otro recurso.
* <p>: define una parte que debe mostrarse como un párrafo.
* <br>: representa un salto de línea.
* <em>: representa un texto enfatizado.
* <hr>: representa un quiebre temático entre párrafos de una sección o articulo o cualquier otro contenido.
* <h1> a <h6>: los elementos de cabecera implementan seis niveles de cabeceras de documentos: <h1> es la mayor y <h6> es la menor importancia. Un elemento de cabecera describe brevemente el tema de la sección que introduce.
* <li>: define un artículo de una lista enumerada.
* <ol>: define una lista ordenada de artículos.
* <td>: representa una celda de datos en una tabla.
* <th>: representa una celda encabezado en una tabla.
* <tr>: representa una fila de celdas en una tabla.
* <ul>: define una lista de artículos sin orden.
* <div>: representa un contenedor genérico sin ningún significado especial.
* <img>: representa una imagen.
* <head>: representa una colección de metadatos acerca del documento, incluyendo enlaces a, o definiciones de scripts y hojas de estilo.
* <body>: representa el contenido principal de un documento HTML. Solo hay un elemento <body> en un documento.
* <meta>: define los metadatos que no pueden ser definidos usando otro elemento HTML.
* <mark>: representa texto resaltado con propósitos de referencia, es decir, por su relevancia en otro contexto.
* <table>: representa datos con más de una dimensión.
* <title>: define el título del documento, el cual se muestra en la barra de título del navegador o en las pestañas de la página. Solo puede contener texto y cualquier otra etiqueta contenida no será interpretada.
* <thead>: representa el bloque de filas que describen las etiquetas de columna de una tabla.
* <tbody>: representa el bloque de filas que describen los datos concretos de una tabla.
* <tfoot>: representa los bloques de filas que describen los resúmenes de una columna de una tabla.
* <strong>: representa un texto importante.
* <caption>: representa el título de una tabla.
* <section>: define en una sección en un documento.

**Atributos generales:**

Todos las etiquetas pueden tener los siguientes atributos generales:

* Clase: establece la clase CSS que se aplica a los estilos del elemento.
* Id: establece un indicador único a cada elemento.

**Atributos específicos:**

Solo las siguientes etiquetas podrán tener atributos específicos:

* <html>:
  + Lang: indica el idioma del elemento.
* <a>:
  + Href: la dirección del recurso enlazado. Es el corazón del vínculo, e indica a los navegadores a donde deben dirigirse cuando el vínculo este activado.
  + Target: un valor que especifica donde debería abrirse el recurso enlazado.
* <ol>:
  + Type: el tipo de símbolos a usar en las viñetas.
* <li>:
  + Value: un entero indicando el valor ordinal del ítem en la lista.
* <img>:
  + Src: el URI del recurso de imagen. Este atributo es obligatorio ya que apunta al recurso que será incrustado en el documento.
  + Alt: una línea de texto que provee contenido equivalente para las situaciones en donde las imágenes no pueden ser vistas.
  + Height: un numero de pixeles que indican la altura que la imagen tomara cuando sea incrustada en el documento.
  + Width: un numero de pixeles que indica el ancho que la imagen tomara cuando sea incrustada en el documento.
* <table>:
  + Border: indica el contorno o borde la etiqueta <table>.

# **3. Analizador léxico:**

Para la segunda etapa del trabajo practico integrador consiste en poder generar un analizador léxico (lexer) el cual permite procesar el lenguaje fuente como una secuencia de caracteres agrupándolas en palabras que tienen un significado propio y luego busca los componentes léxicos que componen el programa según reglas o patrones.

En el cual se tuvo que tener en cuenta:

* **Tokens:** es una unidad léxica indivisible con significado único dentro del lenguaje.
* **Patrones:** es una expresión regular que permite identificar unívocamente un tipo de token y referenciar al conjunto de todos los tokens que se ajustan a él.
* **Lexemas:** es una cadena de caracteres que encaja con un patrón que describe un componente léxico, es decir, es como una instancia de un patrón.

Para la generación del analizador léxico se utilizó:

* Flex (versión 2.5.4a): el cual es una herramienta para generar escáneres: programas que reconocen patrones léxicos en un texto
* Mingw (versión 6.3.0): es una implementación de los compiladores GCC para la plataforma Win32, que permite migrar la capacidad de este compilador en entornos Windows.
* Sistema operativo: Windows 10.
* Lenguaje de programación: C.

# **4. Analizador Sintáctico:**

Para la tercera y ultima entrega del trabajo practico integrador consiste en crear un analizador sintáctico (parser) que analice una cadena de símbolos de acuerdo a como este hecho la gramática.

El análisis sintáctico convierte el texto de entrada en otras estructuras que son más útiles para el posterior análisis y capturan la jerarquía implícita de la entrada. Un analizador léxico crea tokens de una secuencia de caracteres de entrada y son estos tokens los que son procesados por el analizador sintáctico para construir la estructura de datos.

Para la generación del analizador sintáctico se utilizó:

* Bison (versión 2.41): es un generador de analizador de propósito general que convierte una gramática anotada libre de contexto en un analizador determinista LR o generalizado LR (GLR) que emplea tablas de analizador LALR.
* Mingw (versión 6.3.0): es una implementación de los compiladores GCC para la plataforma Win32, que permite migrar la capacidad de este compilador en entornos Windows.
* Sistema operativo: Windows 10.
* Lenguaje de programación: C.

# **5. Flex y Bison:**

**Flex:**

Es una herramienta para generar escáneres: programas que reconocen patrones léxicos en un texto. Flex lee los ficheros de entrada dados, o la entrada estándar si no se le ha indicado ningún nombre de fichero, con la descripción de un escáner a generar. La descripción se encuentra en forma de parejas de expresiones regulares y código C, denominadas reglas. Flex genera como salida un fichero fuente en C, “lex.yy.c”, que define una rutina “yylex()”. Este fichero se compila y se enlaza con la librería ‘-lfl’ para producir un ejecutable. Cuando se arranca el fichero ejecutable, este analiza su entrada en busca de casos de las expresiones regulares. Siempre que encuentra uno, ejecuta el código C correspondiente

Para la creación del scanner se requiere un fichero de entrada (archivo con extensión .l) que se llama a ejecutar desde Flex, está compuesto de tres secciones, separadas por una línea donde aparece únicamente un ‘%%’ en esta:

definiciones

%%

reglas

%%

código de usuario

Características de la secciones:

* Definiciones: es donde se encuentran las declaraciones de definiciones de nombres sencillas para simplificar la especificación del escáner, y declaraciones de condiciones de arranque.
* Reglas: contiene una serie de reglas de la forma: expresión regular – acción.
* Código de usuario: es donde el usuario va a poder generar código de C y este es copiado al archivo fuente llamado “lex.yy.c”. Esta sección se utiliza para rutinas de complemento que llaman al escáner o son llamadas por este.

**Bison:**

Es un generador de analizadores sintácticos de propósito general que convierte una descripción para una gramática independiente del contexto (en realidad de una subclase de éstas, las LALR) en un programa en C que analiza esa gramática. Un fuente de Bison (archivo con extensión .y) describe una gramática. El ejecutable que se genera indica si un fichero de entrada dado pertenece o no al lenguaje generado por esa gramática. La forma general de una gramática de Bison es la siguiente:

%{

Librerías y declaraciones en C

%}

Declaraciones de Bison

%%

Reglas gramaticales

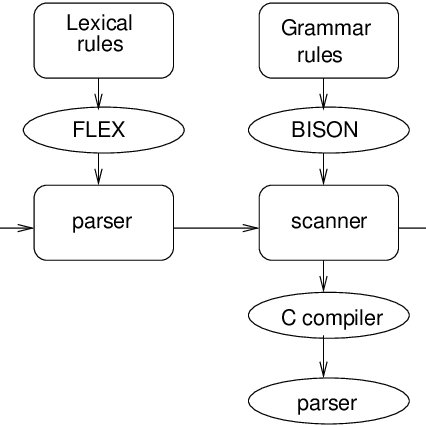
%%

Código C

* librerías y declaraciones en C: se pueden definir tipos y variables utilizadas en las acciones, así como también utilizar #include para incluir archivos de cabecera.
* Declaraciones de Bison: es donde se declaran los nombres de los símbolos terminales y no terminales, y también podrían describir la precedencia de operadores y los tipos de datos de los valores semánticos de varios símbolos.
* Reglas gramaticales: son las producciones de la gramática, que además pueden llevar asociadas acciones, código en C, que se ejecutan cuando el analizador encuentra las reglas correspondientes.
* Código C: es donde se encuentra la función main() para generar el menú del programa.

**Flex-Bison:**

El analizador Bison utiliza la función “yylex()” para devolverle el siguiente token de la entrada. Esa función devuelve el tipo del próximo token y además puede poner cualquier valor asociado en la variable global yylval. Para usar Flex con Bison se genera el fichero “y.tab.h” en el cual se encuentran las definiciones de todos los “%tokens” que aparecen el archivo fuente de Bison. Este fichero de cabecera se incluye después en el fichero fuente de Flex.



# **6. Funciones y librerías utilizadas:**

Herramientas de Flex y Bison:

* yyin(): puntero del flujo de entrada, la entrada predeterminada de main () es stdin.
* yyparse(): devuelve 0 si la entrada que analiza es válida de acuerdo con las reglas gramaticales dadas. Devuelve 1 si la entrada es incorrecta y la recuperación de errores es imposible.
* yylex(): implica el punto de entrada principal para lex, lee el flujo de entrada genera tokens, devuelve cero al final del flujo de entrada.
* yytext(): búfer que contiene los caracteres de entrada que realmente coinciden con el patrón (es decir, lexema).
* yylval(): contiene el valor del token.
* yywrap(): es llamado por lex cuando la entrada está agotada (o en final de archivo).
* yyerror(): función invocada desde yyparse cuando ocurre un error sintáctico.
* yylineno():ordena a flex a generar un analizador que mantenga el número de la línea.
* yydebug 1: es una función de rastreo de depuración, cada paso que realiza el analizador cuando yydebug es distinto de cero se evita que se generen por pantalla una línea o dos de información de rastreo

**Librerías de C:**

* **stdio.h**, que significa "**st**andar**d i**nput-**o**utput **h**eader" (cabecera estándar E/S): es el [archivo de cabecera](https://es.wikipedia.org/wiki/Archivo_de_cabecera) que contiene las definiciones de las [macros](https://es.wikipedia.org/wiki/Macro), las [constantes](https://es.wikipedia.org/wiki/Constante_(programaci%C3%B3n)), las declaraciones de funciones de la [biblioteca estándar](https://es.wikipedia.org/wiki/Biblioteca_est%C3%A1ndar_de_C) del [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [C](https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n)) para hacer operaciones, estándar, de entrada y salida, así como la definición de [tipos](https://es.wikipedia.org/wiki/Tipo_de_dato) necesarias para dichas operaciones. El fichero de cabecera stdio.h define las siguientes variables:
  + stdin: es un puntero hacia otroarchivo para obtener una lectura de carácter. El puntero a FILE hace referencia a la entrada estándar, como el teclado. Los tipos de datos definidos en el fichero de cabecera stdio.h son:
    - FILE: estructura que contiene información sobre el fichero o stream de texto necesario para realizar las operaciones de entrada/salida sobre él. incluye:
      * posición actual de stream.
      * puntero al buffer del stream.
      * indicador de fin de fichero (EOF).
      * indicador de error.
* stdlib.h (std-lib: standard library o biblioteca estándar): es el archivo de cabecera de la biblioteca estándar de propósito general del lenguaje de programación C. Contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras.

**Funciones:**

* printf(): es una función para imprimir por pantalla los datos.
* scanf(): es una función para ingresar datos.
* getch(): es una función que lee carácter a carácter pero no lo muestra por pantalla.
* fopen(): es una función que abre un archivo para lectura/escritura y adición.
* fclose(): es una función que se utiliza luego de leer o escribir un archivo. Es necesario cerrar el archivo para que no haya ningún error luego de la escritura o lectura.
* system(“cls”): es una función que limpia la pantalla.

Las demás funciones que no se mencionan son parte del código que se utilizó para generar el analizador léxico y sintáctico.

# **7. Consideraciones:**

A realizar el analizador léxico y sintáctico se tuvo las siguientes consideraciones:

1. No se pueden utilizar los símbolos “<” y “>” dentro del token llamado Palabra, ya que estos generaban error ahora hora de reconocer las demás etiquetas.
2. Los caracteres especiales tales como tildes, ñ, =, ¡, etc., se tienen en cuentan a la hora de aceptarlos como parte de texto.
3. El analizador no considera las mayúsculas y minúsculas.

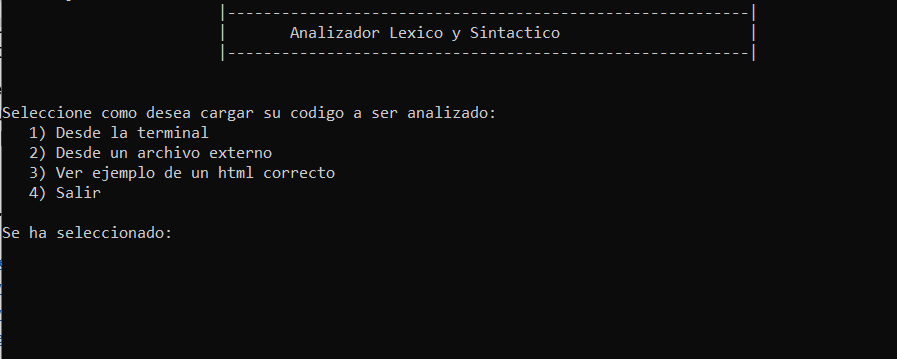
# **8. Ejecución del analizador léxico y sintáctico:**

El analizador posee dos opciones para que el usuario puede analizar el lenguaje de marcado HTML:

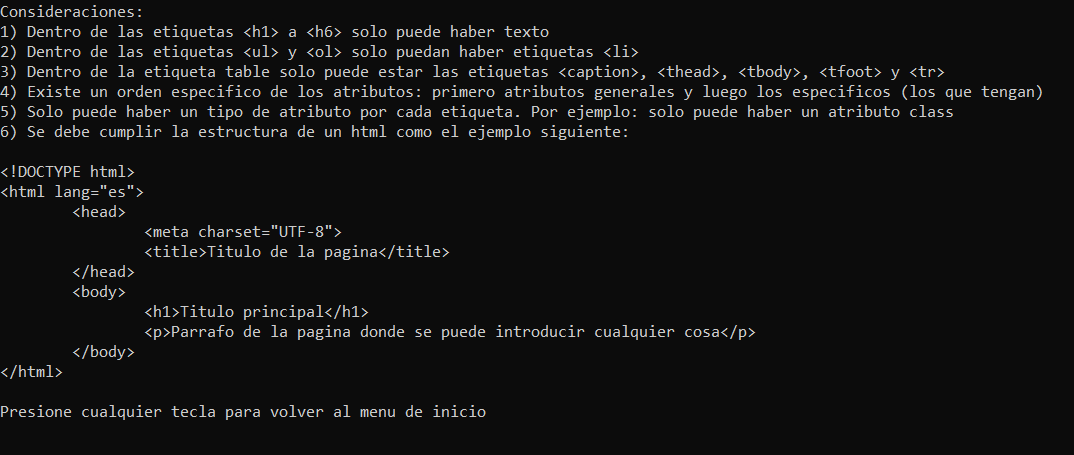
1. Mediante una consola interactiva donde puede ingresar manualmente el código que conforma su HTML.
2. Mediante un archivo ya existente donde se encuentre el código del HTML.

A su vez, posee una opción donde podrá ver las consideraciones que se tiene en cuenta para que su código de HTML sea aceptado por el analizador, y un ejemplo de dicho código.

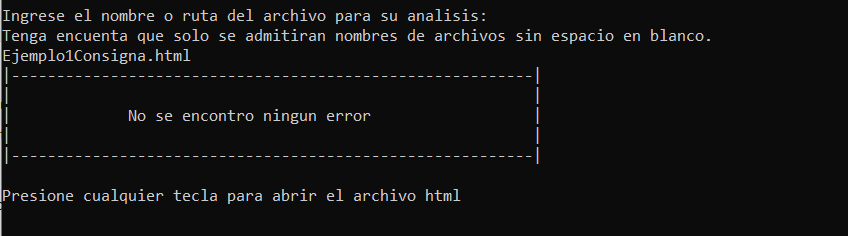
**Menú principal del analizador léxico y sintáctico:**



**Consideraciones y ejemplo:**



**Código HTML aceptado:**



# **9. Bibliografía:**

* <https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5_lista_elementos>
* <https://www.arkaitzgarro.com/xhtml/capitulo-3.html>
* <https://www2.cs.arizona.edu/~debray/Teaching/CSc453/DOCS/tutorial-large.pdf>
* <http://webdiis.unizar.es/~ezpeleta/lib/exe/fetch.php?media=misdatos:compi:2bis.introflex.pdf>
* <https://www.htmlquick.com/es/reference/uri-url.html>
* <https://www.quora.com/What-is-the-function-of-yylex-yyin-yyout-and-fclose-yyout-in-LEX>
* <https://es.wikipedia.org/wiki/GNU_Bison>
* <https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)>
* <https://www.youtube.com/watch?v=gnSok6mbvug>