

Facultad Regional Resistencia

**INGENIERÍA EN SISTEMAS DE INFORMACIÓN**

Trabajo Práctico Integrador: Lexer del Lenguaje HTML (2da entrega)

Materia: Sintaxis y Semántica del Lenguaje - Primer Cuatrimestre

Integrantes del Grupo:

* Pacheco Altamirano Federico
* Samuel Espinola Oscar
* Lucas Emanuel Nuñez
* Cristian Daniel Galarza

Grupo N°: 22.1

Lugar y Fecha: Resistencia, 30 de Mayo de 2020.

Curso Académico 2020

Índice

Introducción……………………………………………………………………………...2

Lenguaje HTML ………………………………………………………………………....3

Características del Lenguaje…………………………………………………...3

Componentes léxicos o tokens………………………………………………...4

Etiquetas para estructura del documento …………………………....4

Etiquetas básicas de párrafo……………………………………….…..5

Listas…………………………………..……………………………….…5

Tablas ……………………………………………………………….……6

Enlaces …………………………………………………………..………7

Imágenes ……………………………………………………..…………8

Atributos globales ………………………………………………………8

Atributos específicos …………………………………………..………8

Texto a mostrar………………………………………………...……….9

Consideraciones…………………………………………………………….....9

Gramática del Lenguaje HTML……….. ………………………………………….…10

Símbolos Terminales y No Terminales……………………………………...10

Tokens………………………………………………………………………….11

Reglas de la Producción………………………………………………...…...12

Análisis Léxico……………………..…………………………………………….…….15

Modo de ejecución del Interprete...…………………………………………….….....21

Anexo…………………………………………………………………………………...23

Diseño e Implementación de Lexer y Parser

**Introducción**

Este documento describe sobre el Trabajo Práctico Integrador: Diseño e Implementación de Lexer y Parser para el lenguaje HTML, realizado por los integrantes del grupo. El mismo se hará en entregas en tres instancias. Donde se detalla en primera instancia información del intérprete: los componentes léxicos (tokens) y sentencias que nuestro Parser y Lexer interpretara del lenguaje HTML, basado en las directivas brindadas por la cátedra respecto a las características del lenguaje sobre el que trabajaremos. En la segunda instancia, consistirá en la presentación del analizador lexicográfico desarrollado, que se encarga de buscar los componentes léxicos (tokens) o palabras que componen el Lenguaje HTML, según unas reglas o patrones. Y por último se desarrollarla el analizador sintáctico que se encargara de comprobar que el programa fuente, además de cumplir con las reglas sintácticas de la gramática diseñada, también cumple las reglas semánticas; esto lo hará con la secuencia de tokens que recibirá el analizador lexicográfico.

Con lo anterior ya definido, aplicando los conocimientos adquiridos en la teoría y la práctica de la materia Sintaxis y Semántica del Lenguaje, presentamos una gramática del tipo 2 (Libre de Contexto) diseñado sobre el Lenguaje HTML, que lo representara. Consistiendo esto en la primera entrega de Trabajo Practico.

Los integrantes del grupo optamos como herramienta para generar los analizadores

lexer y parser, Flex (para el lexer ) y el Bison (para el parser). Ambas herramientas trabajan con el Lenguaje de Programación C.

En esta segunda instancia de entregas se añadirá el Lexer que se implementará en base a la gramática desarrollada, capaz de analizar código HTML; que devolverá las palabras reservas (tokens) que lea en el archivo con el código fuente a analizar. El Lexer desarrollado en Flex es el que pasara los tokens al Parser en la entrega final.

**Requerimientos para ejecutar y compilar el tp:**

Se trabajó bajo el entorno del sistema operativo Windows 10, utilizando su consola de comandos. Pero tendiendo instaladas las siguientes herramientas:

-Los generadores de analizador Flex y Bison (Ver Anexo pág. 23)

-La aplicación MinGW, conocida como MinGW32, es una implementación de los compiladores GCC para la plataforma Win32, que permite migrar la capacidad de este compilador en entornos Windows (Ver Anexo pág. 25)

-Un editor de texto.

**Lenguaje de Marcas de Hipertexto o HTML**

HTML es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla que corresponde a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto, que podría ser traducido como Lenguaje de Formato de Documentos para Hipertexto.

**Características de lenguaje de HTML**

Un documento HTML contiene básicamente texto. Ahora bien, hay texto que sirve para indicar elementos, describir el contenido y propiedades de la página. Es decir, el texto contiene lo que se denomina etiquetas o tags .

Las etiquetas sirven para delimitar elementos de la página; por ejemplo, un párrafo es un elemento de la página, una tabla también. Incluso hay elementos que contienen otros elementos (las tablas constan de filas y las filas de celdas, por ejemplo).

Las etiquetas son textos encerrados entre los signos de mayor y menor (< >). Cuando un navegador se encuentra un texto así encerrado, entenderá que dentro de los símbolos menor y mayor lo que se indica es el inicio de un elemento.

El inicio del elemento se marca con el nombre del elemento entre los símbolos < y >.

Ejemplo:

<strong>

La etiqueta strong sirve para indicar texto importante (se suele mostrar en negrita)

La mayoría de etiquetas afectan a un determinado texto, el cual estará encerrado por las etiquetas. Por lo tanto, casi siempre, existe una etiqueta de apertura y otra de cierre, que se interpretará como el inicio y fin de un determinado elemento de la página.

El cierre de una etiqueta se marca poniendo el símbolo / , seguido del nombre del elemento que estamos cerrando. Ejemplo:

<p>Texto normal <strong>texto negrita</strong></p>

Lo que indicamos en el ejemplo anterior es que un párrafo de texto contiene un elemento de tipo strong, que contiene el texto: texto negrita. El navegador reaccionará mostrando ese texto con un tipo de letra más fuerte. Es decir:

Texto normal **texto negrita**

Ya se ha comentado que un elemento puede contener dentro más elementos, por lo que entre la apertura y el cierre de una etiqueta puede haber más etiquetas. Una norma importante es que siempre debemos de cerrar primero las etiquetas abiertas más tarde.

Es decir, **este código no es correcto**.

Texto normal <strong><em>texto negrita y cursiva </strong></em>

**Atributos**

Algunas etiquetas tienen atributos. Los atributos son propiedades de cada elemento a las que podemos asignar un valor, de modo que dicho valor varía el comportamiento del elemento. Estos atributos pueden definir, por ejemplo, la dirección del texto o el lenguaje usado dentro del elemento o información consultiva para su correcta interpretación, entre muchas otras.

La forma de utilizar atributos es:

<elemento nombreDeAtributo1=”valor1” nombreDeAtributo2=”valor1” … >

A los atributos se les asignan valores que deben ir entre comillas dobles . El símbolo de igualdad (=) es obligatorio.

Ejemplo:

<p lang=”es”>

**Componentes léxicos o tokens. Estructura básica de un documento HTML**

Etiquetas para estructura del documento

Para crear una página web usando HTML 5 la estructura básica es la siguiente:

<!DOCTYPE html>

<html lang=”es”>

<head>

<meta charset=”UTF-8” >

<title></title>

</head>

<body>

<p>cuerpo de la página</p>

</body>

</html>

Donde el significado de los elementos es el siguiente:

* La etiqueta DOCTYPE indica el tipo de HTML que estamos utilizando (concretamente HTML 5).
* La etiqueta html . Todo documento HTML está encerrado dentro de la apertura y el cierre de esa etiqueta. Marca el principio y fin del mismo (marca el elemento raíz de un documento HTML).
  + Se le debe añadir el atributo lang para indicar el idioma en el que está escrito el documento.
* La etiqueta head: Encierra las etiquetas de cabecera, las cuales proporcionan información que el navegador debe tener en cuenta para el contenido del documento aparezca de forma correcta y también contiene elementos que proporcionan información a los buscadores. Se encuentra entre la etiqueta *html* y la etiqueta *body*. Los elementos habituales que contiene son:
  + Las etiquetas meta: Que sirven para indicar aspectos de funcionamiento de la página web como, cabeceras del protocolo http, indicaciones sobre cada cuánto caduca la página, palabras clave para buscar la página en los navegadores, codificación de caracteres de la página,… En este caso sólo se usa la etiqueta meta obligatoria que indica que estamos codificando el texto usando el formato utf-8 de Unicode.

<meta charset="UTF-8">

<meta name="description" content="enunciado interprete lenguaje">

<meta name="keywords" content="HTML,tags,sintaxis">

<meta name="author" content="SSL UTN FFRe">

<meta name="robots" content="index,nofollow" />

* + La etiqueta title . El título de la página, que los navegadores suelen colocar en la barra de título de la ventana en la que se muestra la página.
* El cuerpo. Encerrado en la etiqueta body , encierra el contenido en sí de la página web. Lo que el navegador muestra por pantalla es lo que se coloca en este apartado.
* Espacios en blanco: El texto dentro de las páginas web no respeta los espacios en blanco ni tabuladores que coloquemos en el código, a la hora de mostrar el contenido por pantalla. Solo se considera el primer espacio en blanco, el resto se eliminan.
* section : Define una sección dentro del documento, puede contener los mismos tags que la sección body.
* div : Contenedor genérico para bloques de texto. se pueden colocar todo tipo de etiquetas (tablas, párrafos, imágenes,…).

**Etiquetas básicas de párrafo**

Todo texto debe ir dentro de una etiqueta que indique que tipo de texto es :

* Etiqueta **p** : todo texto dentro de etiquetas p es considerado párrafo normal.
* Etiquetas **h1** a **h6** : texto dentro de estas etiquetas es considerado títulos. h1 es de primer nivel, h2 de segundo, etc.
* Etiqueta **br** : realiza un salto de línea, este elemento no tiene cierre.
* Etiqueta **hr** : similar al anterior , además de salto de línea , dibuja línea horizontal, no necesita cierre.
* Etiqueta **strong** : Texto importante, énfasis fuerte.
* Etiqueta **em** : texto enfatizado, normalmente se muestra en cursiva.
* Etiqueta **mark** : Texto resaltado. sobre el texto se aplica un fondo vivo, similar a marcador.

**Listas**

Las listas permiten crear párrafos agrupados y alineados mediante símbolos como viñetas o números para facilitar la lectura y organización.

* Etiqueta **ul** : engloba listas con viñetas, no ordenada.
* Etiqueta **ol** : engloba listas numéricas, ordenadas.
* Etiqueta l**i** : ítem dentro de la lista.

Observación : Es posible meter una lista dentro de otra, incluso mezclando.

<p> Lista de la compra </p>

<ol>

<li>

No alcohólicas

<ul>

<li> Agua </li>

</ul>

</li>

<li>

Alcohólicas

<ul>

<li> Vino </li>

<li> Cerveza </li>

</ul>

</li>

</ol>

**Tablas**

Una tabla es un conjunto de datos presentados en forma de filas y columnas.

Cada tabla está asociado a un elemento table , dentro de este elemento se indican las filas mediante el elemento tr y dentro de cada fila se indican las celdas mediante elementos td.

<table>

<tr>

<th> &nbsp; </th>

<th>Jueves</th>

<th> Viernes </th>

</tr>

<tr>

<th> 18:30 </th>

<td>Practica Sintaxis</td>

<td> Fisica </td>

</tr>

<tr>

<th> 21:30 </th>

<td> Practica Sintaxis </td>

<td> Mat Discreta </td>

</tr>

</table>

* Celdas de cabecera: Es muy habitual que las tablas muestran datos y que estos posean celdas que sirvan para describirles. Esas celdas se consideran de cabecera y se marcan con th .
* La agrupación de filas: Hay tres elementos HTML que sirven para diferenciar las tres partes principales de una tabla, son:
  + **thead**. Sirve para indicar las filas que forman la cabecera de la tabla
  + **tfoot**. Indica el pie de la tabla
  + **tbody**. Indica el cuerpo de la tabla

<table border=”1” rules=”groups” >

<caption> Ventas por secciones </caption>

<thead>

<tr>

<td> &nbsp ;</td>

<td> Hardware </td>

<td> Software </td>

</tr>

</thead>

<tfoot>

<tr>

<th> Total </th>

<th> 25000 </th>

<th> 22000 </th>

</tr>

</tfoot>

<tbody>

<tr>

<th> Enero </th>

<td> 12000 </td>

<td> 15000 </td>

</tr>

<tr>

<th> Febrero </th>

<td> 13000 </td>

<td> 9000 </td>

</tr>

</tbody>

</table>

**Enlaces**

La etiqueta de enlaces permiten colocar un texto (u otro elemento, como una imagen) resaltado de forma especial, de modo que cuando se le hace clic, nos llevará al destino de la URL.

* Etiqueta **a**: crea enlaces a texto u otros recursos. El atributo **href** permite indicar la URL a la que se realiza el salto.
* Los enlaces pueden ser absolutos, locales relativos, o internos dentro del propio documento.

Nota: Una URL ( Uniform Resource Location ) es la dirección concreta de un recurso en Internet.

La sintaxis completa de una URL es la siguiente:

protocolo://dominio:puerto/ruta#fragmento

Se detalla el significado de todos los elementos:

● El **protocolo o esquema de red** . Hace referencia al nombre de protocolo de red necesario para poder alcanzar el recurso al que hace referencia la URL. Protocolos habituales son:

○ **http://** (para recursos de la web)

○ **https://** (para recursos de la web contenidos en un servidor seguro)

○ **ftp://** (recursos contenidos en un servidor de ficheros)

○ **ftps://** (recursos contenidos en un servidor de ficheros seguro)

Tras el protocolo se indican dos puntos, tras los cuales normalmente se indican dos barras para indicar la máquina.

● **Nombre de dominio** . Nombre completo en Internet de la máquina (o la red) que posee el recurso en forma de nombre de dominio.no distinguen entre mayúsculas y

minúsculas.

● **Puerto.** Opcional. Puerto por el que se debe conectar con el servidor para obtener el recurso. Si no se indica (que es lo habitual) se toma el puerto por defecto. Por ejemplo en http el puerto por defecto es el 80 . Si queremos usar uno en particular se indica tras el servidor poniendo dos puntos y el puerto.

● **Ruta.** Opcional. Indica el recorrido dentro de la máquina remota que hay que hacer a través de los directorios para llegar al recurso que queremos. Se pone después del servidor. Ejemplos:

○ /index.html Accede a la página index.html situada en el directorio raíz.

○ /imagenes/paisajes/foto001.jpg Accede a la imagen foto001.jpg dentro del directorio paisajes dentro, a su vez, del directorio imagenes.

● **Localizador interno**. Opcional . Va detrás del símbolo **#** y sirve para indicar un

identificador (que puede ser más o menos complejos) que permita localizar o

seleccionar una parte concreta del recurso destino de la URL.

● Los únicos caracteres permitidos en URL son letras, números, guión medio, guión bajo y punto. además de los caracteres reservados: **#** , **/**, **:**

**Imágenes**

Las imágenes se colocan mediante la etiqueta img . El atributo src indica la URL (relativa o absoluta) a la imagen. Ejemplo:

<img src=”imagenes/boton.gif” >

**img:** es una etiqueta sin cierre, la imagen se coloca directamente en la posición de esta etiqueta. Se la considera como texto al maquetarla. De modo que aparece entre el texto si ponemos la etiqueta entre el texto.

**Atributos globales**

* **CLASS**: El nombre de una clase, o una lista de nombres de clases separados por espacios. Este atributo es utilizado en conjunto con hojas de estilo e indica al navegador qué clases se supone asignarán atributos presentacionales al elemento.
  + Ejemplo de uso: <p class="referencias"> lorem ipsum </p>
* **ID**: Un identificador para el elemento. Este identificador debe ser único en el documento y puede ser utilizado para referirse al elemento desde vínculos, scripts, definiciones de estilo u otros atributos.
  + Ejemplo de uso: <h1 id=”clave”> lorem ipsum </h1>

**Atributos específicos**

1. **ELEMENTO <ol>**:
   1. **TYPE**: El tipo de símbolos a usar en las viñetas. Existen cinco posibles valores:
      1. 1: enteros (1, 2, 3...).
      2. a: letras minúsculas (a, b, c...).
      3. A: letras mayúsculas (A, B, C...).
      4. i: números romanos en minúsculas (i, ii, iii...).
      5. I: números romanos en mayúsculas (I, II, III...).
2. **ELEMENTO <li>**:
   1. **VALUE**: Un entero indicando el valor ordinal del ítem en la lista.
3. **ELEMENTO <a>**:
   1. **HREF**: La dirección (representada por un URI) del recurso enlazado. Este atributo es el corazón del vínculo, e indica a los navegadores a dónde deben dirigirse cuando el vínculo es activado.
   2. **TARGET**: Un valor que especifica dónde debería abrirse el recurso enlazado.
      1. \_blank: el vínculo se abrirá en una nueva ventana.
      2. \_parent: el vínculo se abrirá en el contexto padre inmediato.
      3. \_self: el vínculo se abrirá en el mismo contexto que contiene al vínculo.
      4. \_top: el vínculo se abrirá en el más alto contexto de navegación (el contexto más exterior que contiene al vínculo).
4. **ELEMENTO <img>** :
   1. **ALT** Una línea de texto que provee contenido equivalente para aquellas situaciones en que las imágenes no pueden ser vistas (usuarios con discapacidades visuales, navegadores sin soporte, configuraciones en que las imágenes están deshabilitadas, etc.).
   2. **SRC**: El URI del recurso de imagen. Este atributo es obligatorio ya que apunta al recurso que será, de hecho, incrustado en el documento.
   3. **WIDTH**: Un número de píxeles que indica el ancho que la imagen tomará cuando sea incrustada en el documento.
   4. **HEIGHT**: Un número de píxeles que indica la altura que la imagen tomará cuando sea incrustada en el documento.

**Texto a mostrar**

Características

# Estarán compuestos por una serie de letras mayúsculas, minúsculas, dígitos y caracteres especiales. Todos compatibles con el estándar UTF-8.

No se distinguirá entre mayúsculas ni minúsculas.

**Consideraciones:**

1. Restricciones a las etiquetas:
   1. Las etiquetas de header **<h1>** a **<h6>** y demás solo pueden contener texto plano (no otras etiquetas)
   2. Las etiquetas de lista **<ul>** **<ol>** solo contendrán elementos de **item** de lista **<li>**, donde ese item si puede tener cualquier cosa dentro.
   3. Las tablas, contendrán solo la estructura (filas, **theader, tfooter** y **tbody** como se muestra anteriormente) y los elementos de celda **<th>** y **<td>** puedan tener cualquier cosa dentro
   4. <**head**> contendrá solo las meta y el title
   5. <**title**> tiene solo texto
2. Las etiquetas **meta**, es obligatorio que esté **meta chartset** (lo cual obliga que haya un **head**) y además pueden venir otras genéricas (con **name** y **content**). No importa que venga dentro de **name ="ACA"** y dentro de **content="aca"** , siempre que sea solo texto.
3. El atributo **lang** es obligatorio, también considerar que el valor del atributo deben ser dos letras, no importa cuales. Por ejemplo (**lang="en"**, **lang="es"**, **lang="fr"**, **lang="ag"** )
4. Respecto a los atributos, que pueden o no ir :
   1. Globales (posible para todos): CLASS y ID
   2. Específicos, según el elemento (tag):
      1. <ol> TYPE
      2. <li> VALUE
      3. <a> HREF (requerido) y TARGET
      4. <img> ALT SRC(requerido) WITH HEIGHT
   3. Los ejemplificados que también incluímos:

* <html> Lang (requerido)
* <table> BORDER
* <meta> dichas más arriba

**Gramática del Lenguaje HTML**

En esta parte del documento, presentamos la gramática que diseñamos para el Lenguaje HTML.

|  |
| --- |
| Símbolos de la Gramática: V={ } |
| <,>,/, :, =, “, ”, #, !DOCTYPE, html, head, meta, charset, title, body, p, h1, h2, h3, h5, h6, strong, em, mark, br, hr, div, section, ol, ul, li, table, caption, thead, tfoot, tbody, tr, th, td, , img, target, class, id, |

|  |
| --- |
| Símbolos terminales: N={ } |
| <!DOCTYPE html> , </html>, <head>, </head>, <meta>, charset=, <title>, </title>, <body>, </body>, <p>, < /p>, <h1>, </h1>, <h2>, </h2>, <h3>, </h3>, <h4>, </h4>, <h5>, </h5>, <h6>, </h6>, <strong>, </strong>, <em>, </em>, <mark>, </mark>, <hr>, <br>, <div>, </div>, <section>, </section>, <table>, </table>, <a>, </a>, <img>, <ol>, </ol>, <ul>, </ul>, <li>, </li>, <caption>, </caption>, <thead>, </thead>, <tfoot>, </tfoot>, <tbody>, </tbody>, <tr>, </tr>, <th>, </th>, <td>, </td>, class =, id=, {string} , {numero}, {atA}, {atImagen}, {atOl}, {atLi}, {atMeta} |

|  |
| --- |
| Símbolos no terminales: T={ } |
| 𝚺, {contenido}, {apHead}, {cabecera}, {apMeta}, {cieMeta}, {meta}, {metaExtra}, {aptitulo}, {titulo}, {apBody}, {cuerpo}, {apP}, {cieP}, {elemento}, {aph1}, {cieh1}, {aph2}, {cieh2}, {aph3}, {cieh3}, {aph4}, {cieh4}, {aph5}, {cieh5}, {aph6}, {cieh6}, {apStrong}, {cieStrong}, {elemento}, {apEm}, {cieEm}, {apMark}, {cieMark}, {apDiv}, {cieDiv}, {apSection}, {cieSection}, {apTabla}, {cieTabla}, {apA}, {cieA}, {Img}, {apOL}, {apUL}, {cieUL}, {apLI}, {cieLI}, {item}, {apCaption}, {cieCaption}, {eleDeTabla}, {apThead}, {cieThead}, {eleDeTabla}, {apTfoot}, {cieTfoot}, {apTbody}, {cieTbody}, {apTr}, {cieTr}, {filas}, {apTh}, {cieTh}, {celdas}, {apTd}, {cieTd}, {celdas}, {atGloblaes} |

Definición de los Tokens:

* **{string}**: este es un símbolo terminal de la gramática que representa una cadena de texto; una cadena de caracteres, palabras, ristra de caracteres o frase es una secuencia ordenada (de longitud arbitraria, aunque finita) de elementos que pertenecen a un cierto lenguaje formal o alfabeto análogas a una fórmula o a una oración. Consiste en una combinación de caracteres alfanuméricos.
* **{numero}**: este es un símbolo terminal que representa a los dígitos numéricos dentro del Lenguaje HTML

Los siguientes tokens corresponden a los atributos específicos de una etiqueta en particular:

* **{atHtml}**: este es un símbolo terminal de la gramática que representa los tributos específicos de la etiqueta <html>. El atributo es:
  + **lang=** (obligatorio): que contiene como valor solamente dos letras.
* **{atA}**: este es un símbolo terminal de la gramática que representa los tributos específicos de un enlace, el elemento <a>. Los atributos son:
  + **href**= (obligatorio): Este atributó contendrá la url a la que queremos enlazar. Compuesto por los obligatorios protocolo y dominio; los opcionales: puerto, ruta, localizador interno:

“protocolo://dominio:puerto/ruta#fragmento”

* + **target=** (opcional) : este atributo puede contener solo 4 valores posibles
    - **\_blank**
    - **\_parent**
    - **\_self**
    - **\_top**
* **{atImagen}**: este es un símbolo terminal de la gramática que representa los tributos específicos del elemento <img>. Los atributos son:
  + **alt=**”**{string}**” (opcional)
  + **src=**(obligatorio): contendra la URL de la imagen.
  + **width=”{numero}”**(opcional)
  + **height=”{numero}”**(opcional)
* **{atOl}**: este es un símbolo terminal de la gramática que representa los tributos específicos del elemento <ol>. Los atributos son:
  + **type=” ”** (opcional): puede contener lo siguiente:
    - 1: enteros (1, 2, 3...).
    - a: letras minúsculas (a, b, c...).
    - A: letras mayúsculas (A, B, C...).
    - i: números romanos en minúsculas (i, ii, iii...).
    - I: números romanos en mayúsculas (I, II, III...).
* **{atLi}**: este es un símbolo terminal de la gramática que representa los tributos específicos del elemento <li>. Los atributos son:
  + **value=”{numero}”**(opcional)
* **{atMeta}:** este es un símbolo terminal de la gramática que representa los tributos específicos de la etiqutea <meta>. Los atributos son:
  + **name=”{string}”**(opcional)
  + **content=”{string}”**(opcional)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gramatica | | | | |
| **1)** | 𝚺 | | ➔ | <!DOCTYPE html>{contenido} |
| **2)** | {contenido} | | ➔ | <html {atHtml}> {cabecera} {cuerpo} </html> |
| Cabecera | | | | |
| <head> | | | | |
| **3)** | {apHead} | ➔ | | <head {atGlobales}> | <head> |
| **4)** | {cabecera} | ➔ | | {apHead} {meta} </head> |
| **5)** | {cabecera} | ➔ | | {apHead} {meta} {titulo}</head> |
| <meta> | | | | |
| **6)** | {apMeta} | | ➔ | <meta {atGlobales} | <meta |
| **7)** | {cieMeta} | | ➔ | > {metaExtra} | > |
| **8)** | {meta} | | ➔ | {apMeta} charset="{string}"> {cieMeta} |
| **9)** | {meta} | | ➔ | {apMeta} charset="{string}" {atMeta}> {cieMeta} |
| **10)** | {metaExtra} | | ➔ | {apMeta} {atMeta} {cieMeta} |
| <title> | | | | |
| **11)** | {aptitulo} | | ➔ | <title {atGlobales} >| <title> |
| **12)** | {titulo} | | ➔ | {aptitulo} {elemento}</title> |
| Cuerpo | | | | |
| <body> | | | | |
| **13)** | {apBody} | | ➔ | <body {atGlobales} >| <body> |
| **14)** | {cuerpo} | | ➔ | {apBody} {elemento}</body> |
| Tags del body | | | | |
| <p> | | | | |
| **15)** | {apP} | | ➔ | <p {atGlobales}> | < p> |
| **16)** | {cieP} | | ➔ | </p> | </p> {elemento} |
| **17)** | {elemento} | | ➔ | {apP} {elemento} {cieP} |
| **18)** | {elemento} | | ➔ | {apP} {string } {cieP} |
| <h1> | | | | |
| **19)** | {aph1} | | ➔ | <h1 {atGlobales} >| <h1> |
| **20)** | {cieh1} | | ➔ | </h1> | </h1> {elemento} |
| **21)** | {elemento} | | ➔ | {aph1} {string} {cieh1} |
| <h2> | | | | |
| **22)** | {aph2} | | ➔ | <h2 {atGlobales}> | <h2> |
| **23)** | {cieh2} | | ➔ | </h2> | </h2> {elemento} |
| **24)** | {elemento} | | ➔ | {aph2} {string} {cieh2} |
| <h3> | | | | |
| **25)** | {aph3} | | ➔ | <h3 {atGlobales}> | <h3> |
| **26)** | {cieh3} | | ➔ | </h3> | </h3> {elemento} |
| **27)** | {elemento} | | ➔ | {aph3} {string} {cieh3} |
| <h4> | | | | |
| **28)** | {aph4} | | ➔ | <h4 {atGlobales} > | <h4> |
| **29)** | {cieh4} | | ➔ | </h4> | </h4> {elemento} |
| **30)** | {elemento} | | ➔ | {aph4} {string} {cieh4} |
| <h5> | | | | |
| **31)** | {aph5} | | ➔ | <h5 {atGlobales} > | <h5> |
| **32)** | {cieh5} | | ➔ | </h5> | </h5> {elemento} |
| **33)** | {elemento} | | ➔ | {aph5} {string} {cieh5} |
| <h6> | | | | |
| **34)** | {aph6} | | ➔ | <h6 {atGlobales} >| <h6> |
| **35)** | {cieh6} | | ➔ | </h6> | </h6> {elemento} |
| **36)** | {elemento} | | ➔ | {aph6} {string} {cieh6} |
| <Strong> | | | | |
| **37)** | {apStrong} | | ➔ | <strong {atGlobales}> | <strong> |
| **38)** | {cieStrong} | |  | </strong>{elemento} | </strong> |
| **39)** | {elemento} | | ➔ | {apStrong} {string} {cieStrong} |
| <em> | | | | |
| **40)** | {apEm} | | ➔ | <em {atGlobales}> | <em> |
| **41)** | {cieEm} | |  | </em>{elemento} | </em> |
| **42)** | {elemento} | | ➔ | {apEm} {string} {cieEm} |
| <Mark> | | | | |
| **43)** | {apMark} | | ➔ | <mark {atGlobales}> | <mark> |
| **44)** | {cieMark} | |  | </mark>{elemento} | </mark> |
| **45)** | {elemento} | | ➔ | {apMark} {string} {cieMark} |
| <br> | | | | |
| **46)** | {elemento} | | ➔ | <br {atGlobales}> | <br> |
| **47)** | {elemento} | | ➔ | <br {atGlobales}> {elemento} | <br> {elemento} |
| <hr> | | | | |
| **48)** | {elemento} | | ➔ | <hr {atGlobales} > | <hr> |
| **49)** | {elemento} | | ➔ | <hr {atGlobales} > {elemento} |<hr> {elemento} |
| <Div> | | | | |
| **50)** | {apDiv} | | ➔ | <div {atGlobales} > | <div> |
| **51)** | {cieDiv} | | ➔ | </div>{elemento} |</div> |
| **52)** | {elemento} | | ➔ | {apDiv} {elemento} {cieDiv} |
| **53)** | {elemento} | | ➔ | {apDiv} {string} {cieDiv} |
| <Section> | | | | |
| **54)** | {apSection} | | ➔ | <section {atGlobales} > |<section> |
| **55)** | {cieSection} | | ➔ | </section>{elemento} |</section> |
| **56)** | {elemento} | | ➔ | {apSection} {string} {cieSection} |
| **57)** | {elemento} | | ➔ | {apSection} {elemento} {cieSection} |
| <table> | | | | |
| **58)** | {apTabla} | | ➔ | <table {atGlobales} {atTabla} >| <table {atGlobales} >| <table {atTabla} >|<table> |
| **59)** | {cieTabla} | | ➔ | </table> |</table> {elemento} |
| **60)** | {elemento} | | ➔ | {apTabla} {eleDeTabla} {cieTabla} |
| <a> | | | | |
| **61)** | {apA} | | ➔ | <a {atGlobales} {atA}> | <a {atA}> |
| **62)** | {cieA} | | ➔ | </a> | </a> {elemento} |
| **63)** | {elemento} | | ➔ | {apA} {string} {cieA} |
| <img> | | | | |
| **64)** | {Img} | | ➔ | <Img {atImgen}> | <Img {atGlobales} {atImgen}> |
| <ol> | | | | |
| **65)** | {apOL} | | ➔ | <ol{atGlobales} {atOl} >|<ol {atOl} >|<ol{atGlobales}> |<ol> |
| **66)** | {elemento} | | ➔ | {apOL} {item} </ol> |
| <ul> | | | | |
| **67)** | {apUL} | | ➔ | <ul {atGlobales}> | <ul> |
| **68)** | {cieUL} | | ➔ | </ul>{elemento} | </ul> |
| **69)** | {elemento} | | ➔ | {apUL} {item} {cieUL} |
| <li> | | | | |
| **70)** | {apLI} | | ➔ | <li {atGlobales} {atli} >| <li {atGlobales} >| <li {atli} >| <li> |
| **71)** | {cieLI} | | ➔ | </li>{item} | </li> |
| **72)** | {item} | | ➔ | {apLI} {string} {cieLI} |
| **73)** | {item} | | ➔ | {apLI} {elemento} {cieLI} |
| Elemento de la tabla | | | | |
| <caption> | | | | |
| **74)** | {apCaption} | | ➔ | <caption {atGlobales}> | <caption> |
| **75)** | {cieCaption} | | ➔ | </caption>{eleDeTabla} | </caption> |
| **76)** | {eleDeTabla} | | ➔ | {apCaption} {string} {cieCaption} |
| <thead> | | | | |
| **77)** | {apThead} | | ➔ | <thead {atGlobales} >| <thead> |
| **78)** | {cieThead} | |  | </thead>{eleDeTabla} | </thead> |
| **79)** | {eleDeTabla} | | ➔ | {apThead} {filas} {cieThead} |
| <tfoot> | | | | |
| **80)** | {apTfoot} | | ➔ | <tfoot {atGlobales} >| <tfoot> |
| **81)** | {cieTfoot} | |  | </tfoot>{eleDeTabla} | </tfoot> |
| **82)** | {eleDeTabla} | | ➔ | {apTfoot} {filas} {cieTfoot} |
| <tbody> | | | | |
| **83)** | {apTbody} | | ➔ | <tbody {atGlobales} >| <tbody> |
| **84)** | {cieTbody} | |  | </tbody>{eleDeTabla} | </tbody> |
| **85)** | {eleDeTabla} | | ➔ | {apTbody} {filas} {cieTbody} |
| <tr> | | | | |
| **86)** | {apTr} | | ➔ | <tr {atGlobales}> |<tr> |
| **87)** | {cieTr} | | ➔ | </tr>{eleDeTabla} |</tr>{filas} | </tr> |
| **88)** | {filas} | | ➔ | {apTr} {celdas} {cieTr} |
| <th> | | | | |
| **89)** | {apTh} | | ➔ | <th {atGlobales} > | <th> |
| **90)** | {cieTh} | | ➔ | </th> {celdas} |</th> |
| **91)** | {celdas} | | ➔ | {apTh} {string} {cieTh} |
| **92)** | {celdas} | | ➔ | {apTh} {elemento} {cieTh} |
| <td> | | | | |
| **93)** | {apTd} | | ➔ | <td {atGlobales} > | <td> |
| **94)** | {cieTd} | | ➔ | </td> {celdas} |</td> |
| **95)** | {celdas} | | ➔ | {apTd} {string} {cieTd} |
| **96)** | {celdas} | | ➔ | {apTd} {elemento} {cieTd} |
| Atributos Globlales | | | | |
| **97)** | {atGloblaes} | | ➔ | id="string" |
| **98)** | {atGloblaes} | | ➔ | id="string" class="string" |
| **99)** | {atGloblaes} | | ➔ | class="string" |
| **100)** | {atGloblaes} | | ➔ | class="string"id="string" |

**Análisis Léxico**

Teniendo la gramática libre de contexto ya presentada. En esta parte del documento se encuentra el Lexer desarrollado por medio de la herramienta Flex. Este lexer reconocerá e informara los componentes léxicos que encuentra. El mismo está dividido en tres secciones:

**Sección de Declaraciones o Definiciones:**

En la sección de definiciones contiene declaraciones de definiciones de nombres simples y las expresiones regulares correspondientes para simplificar la especificación del analizador, y declaraciones de condiciones de inicio.

%option noyywrap

%option caseless

%{

#include <stdio.h>

int linea,op;

%}

/\*TOK GENERALES\*/

Parrafo [a-zA-Z|á|é|í|ó|ú|ñ|.|=|:|\/||\"|,|\-|\_|?|¿|")"|"("|"#"|"\*"|0-9]+

Letra [a-z A-Z]

Numero [0-9]+

espacios [ \t\n]+

/\*ATRIBUTOS globales\*/

atGlobales (({espacios}id=\"{Parrafo}\")?({espacios}class=\"{Parrafo}\")? |

{espacios}class=\"{Parrafo}\")?({espacios}id=\"{Parrafo}\")?)

/\*ATRIBUTOS html\*/

atHtml (({espacios}lang=\"{Letra}{2}\")?)

/\*ATRIBUTOS de meta\*/

atmeta (({espacios}charset=\"{Parrafo}\") | ({espacios} charset=\"{Parrafo}\")

({espacios}name=\"{Parrafo}\")?({espacios}content=\"{Parrafo}\")? | ({espacios}charset=\"{Parrafo}\")({espacios}content=\"{Parrafo}\")? ({espacios}name=\"{Parrafo}\")? | ({espacios}name=\"{Parrafo}\")? ({espacios}content=\"{Parrafo}\")? | ({espacios}content=\"{Parrafo}\")? ({espacios}name=\"{Parrafo}\")?)

/\*ATRIBUTOS a\*/

target (\_blank|\_self|\_top|\_parent)

atA (({atGlobales})?(({espacios}href=\"{enlace}\")?({espacios}target=\"{target}\")? | ({espacios}target=\"{target}\")?)({espacios}href=\"{enlace}\")?)

/\*ATRIBUTOS img\*/

atExtra (({espacios}alt=\"{Parrafo}\")?({espacios}width=\"{Numero}\")?({espacios} height=\"{Numero}\")? | ({espacios}width=\"{Numero}\")? ({espacios}height=\"{Numero}\")?({espacios}alt=\"{Parrafo}\")? | ({espacios}width=\"{Numero}\")?({espacios}alt=\"{Parrafo}\")?({espacios}height=\"{Numero}\")? | ({espacios}height=\"{Numero}\")? ({espacios}alt=\"{Parrafo}\")?({espacios}width=\"{Numero}\")? | ({espacios}height=\"{Numero}\")?({espacios}width=\"{Numero}\")?({espacios}alt=\"{Parrafo}\")? | ({espacios}alt=\"{Parrafo}\")? ({espacios}height=\"{Numero}\")?({espacios}width=\"{Numero}\")?)

atImg (({atGlobales})?({espacios}src=\"{enlace}\")({atExtra})?)

/\*ATRIBUTOS table\*/

atTabla ({atGlobales}?({espacios}border=\"{Numero}\")?)

/\*ATRIBUTOS lista\*/

tipoDeItem [1aAiI]

atOl ({atGlobales}?({espacios}type=\"{tipoDeItem}\")?)

atLi ({atGlobales}?({espacios}value=\"{Numero}\")?)

/\*ATRIBUTOS URL\*/

stringURL [a-z A-Z|0-9|\-|\_|.|\/]+

protocolo (http|https|fttp|fttps)

enlace ({protocolo}:"/""/"{stringURL}(:{Numero})? ("/"{stringURL})? (\#{stringURL})?)

**Sección de Reglas:**

Esta sección contiene reglas con el formato PATRÓN ACCIÓN, donde cada acción es

un fragmento de programa que describe cual ha de ser la acción que realizará el

analizador léxico cuando el patrón concuerde con un lexema del fichero que contiene el código fuente.

%%

(" ")+ {}

[\t]+ {}

\n {linea++;}

<<EOF>> return 0;

"<"{espacios}?\!DOCTYPE{espacios}"html"{espacios}?">"

{printf("Se leyo una etiqueta DOCTYPE: %s \n",yytext);}

/\*HTML\*/

"<"{espacios}?html{atHtml}{espacios}?">" {printf("Se leyo apertura html: %s \n",yytext);}

"<"{espacios}?\/html{espacios}?">" {printf("Se leyo cierre de html: %s \n",yytext);}

/\*CABECERA\*/

"<"{espacios}?head({atGlobales})?{espacios}?">" {printf("Se leyó apertura de head: %s \n",yytext);}

"<"{espacios}?\/head{espacios}?">" {printf("Se leyo cierre de head: %s \n",yytext);}

"<"{espacios}?meta{atmeta}{espacios}?">" {printf("Se leyo etiqueta meta: %s \n",yytext);}

"<"{espacios}?title({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de title: %s \n",yytext);}

"<"{espacios}?\/title{espacios}?">" {printf("Se leyo cierre de title: %s \n",yytext);}

/\*BODY\*/

"<"{espacios}?body({atGlobales})?{espacios}?">" {printf("Se leyo apertura de body: %s \n",yytext);}

"<"{espacios}?\/body{espacios}?">" {printf("Se leyo cierre de body: %s \n",yytext);}

/\*ETIQUETA DE p\*/

"<"{espacios}?p({atGlobales})?{espacios}?">"

{printf("Se leyo una apertura de parrafo: %s \n",yytext);}

"<"{espacios}?\/p{espacios}?">" {printf("Se leyo cierre de parrafo: %s \n",yytext);}

/\*ETIQUETAS h\*/

"<"{espacios}?h1({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de h1: %s \n",yytext);}

"<"{espacios}?\/h1{espacios}?">" {printf("Se leyo cierre de h1: %s \n",yytext);}

"<"{espacios}?h2({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de h2: %s \n",yytext);}

"<"{espacios}?\/h2{espacios}?">" {printf("Se leyo cierre de h2: %s \n",yytext);}

"<"{espacios}?h3({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de h3: %s \n",yytext);}

"<"{espacios}?\/h3{espacios}?">" {printf("Se leyo cierre de h3: %s \n",yytext);}

"<"{espacios}?h4({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de h4: %s \n",yytext);}

"<"{espacios}?\/h4{espacios}?">" {printf("Se leyo cierre de h4: %s \n",yytext);}

"<"{espacios}?h5({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de h5: %s \n",yytext);}

"<"{espacios}?\/h5{espacios}?">" {printf("Se leyo cierre de h5: %s \n",yytext);}

"<"{espacios}?h6({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de h6: %s \n",yytext);}

"<"{espacios}?\/h6{espacios}?{espacios}?">" {printf("Se leyo cierre de h6: %s \n",yytext);}

/\*ETIQUETAS de resalte y salto\*/

"<"{espacios}?strong({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de strong: %s \n",yytext);}

"<"{espacios}?\/strong{espacios}?">" {printf("Se leyo cierre de strong: %s \n",yytext);}

"<"{espacios}?em({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de em: %s \n",yytext);}

"<"{espacios}?\/em{espacios}?">" {printf("Se leyo cierre de em: %s \n",yytext);}

"<"{espacios}?mark({atGlobales})?{espacios}?">" {printf("Se leyo apertura de mark: %s \n",yytext);}

"<"{espacios}?\/mark{espacios}?">" {printf("Se leyo cierre de mark: %s \n",yytext);}

"<"{espacios}?br({atGlobales})?{espacios}?">"

{printf("Se leyo etiqueta br: %s \n",yytext);}

"<"{espacios}?hr({atGlobales})?{espacios}?">"

{printf("Se leyo etiqueta hr: %s \n",yytext);}

/\*ETIQUETA de div y section\*/

"<"{espacios}?div({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de div: %s \n",yytext);}

"<"{espacios}?\/div{espacios}?">" {printf("Se leyo cierre de div: %s \n",yytext);}

"<"{espacios}?section({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de section: %s \n",yytext);}

"<"{espacios}?\/section{espacios}?">" {printf("Se leyo cierre de section: %s \n",yytext);}

/\*ETIQUETA a\*/

"<"{espacios}?a{atA}{espacios}?">" {printf("Se leyo apertura de a: %s \n",yytext);}

"<"{espacios}?\/a{espacios}?">" {printf("Se leyo cierre de a: %s \n",yytext);}

/\*ETIQUETA img\*/

"<"{espacios}?img{atImg}{espacios}?">" {printf("Se leyo etiqueta de imagen: %s \n",yytext);}

/\* ETIQUETA de lista \*/

"<"{espacios}?ul({atGlobales})?{espacios}?">"

{printf("Se leyo apertura de ul: %s \n",yytext);}

"<"{espacios}?\/ul{espacios}?">" {printf("Se leyo cierre de ul: %s \n",yytext);}

"<"{espacios}?ol{atOl}{espacios}?">" {printf("Se leyo apertura ol: %s \n",yytext);}

"<"{espacios}?\/ol{espacios}?">" {printf("Se leyo cierre de ol: %s \n",yytext);}

"<"{espacios}?li({atLi})?{espacios}?">" {printf("Se leyo apertura de li: %s \n",yytext);}

"<"{espacios}?\/li{espacios}?">" {printf("Se leyo cierre de li: %s \n",yytext);}

/\*TABLE\*/

"<"{espacios}?table{atTabla}{espacios}?">" {printf("Se leyo apertura table: %s \n",yytext);}

"<"{espacios}?\/table{espacios}?">" {printf("Se leyo cierre de table: %s \n",yytext);}

"<"{espacios}?caption({atGlobales})?{espacios}?">"

{printf("Se leyo apertura caption: %s \n",yytext);}

"<"{espacios}?\/caption{espacios}?">" {printf("Se leyo cierre de caption: %s \n",yytext);}

"<"{espacios}?thead({atGlobales})?{espacios}?">"

{printf("Se leyo apertura thead: %s \n",yytext);}

"<"{espacios}?\/thead{espacios}?">" {printf("Se leyo cierre de thead: %s \n",yytext);}

"<"{espacios}?tfoot({atGlobales})?{espacios}?">"

{printf("Se leyo apertura tfoot: %s \n",yytext);}

"<"{espacios}?\/tfoot{espacios}?">" {printf("Se leyo cierre de tfoot: %s \n",yytext);}

"<"{espacios}?tbody({atGlobales})?{espacios}?">"

{printf("Se leyo apertura tbody: %s \n",yytext);}

"<"{espacios}?\/tbody{espacios}?">" {printf("Se leyo cierre de tbody: %s \n",yytext);}

"<"{espacios}?tr({atGlobales})?{espacios}?">"

{printf("Se leyo apertura tr: %s \n",yytext);}

"<"{espacios}?\/tr{espacios}?">" {printf("Se leyo cierre de tr: %s \n",yytext);}

"<"{espacios}?th({atGlobales})?{espacios}?">"

{printf("Se leyo apertura th: %s \n",yytext);}

"<"{espacios}?\/th{espacios}?">" {printf("Se leyo cierre de th: %s \n",yytext);}

"<"{espacios}?td({atGlobales})?{espacios}?">"

{printf("Se leyo apertura td: %s \n",yytext);}

"<"{espacios}?\/td{espacios}?">" {printf("Se leyo cierre de td: %s \n",yytext);}

/\*URL\*/

{enlace} {printf("Se leyo un enlace: %s \n",yytext);}

/\*TEXTO\*/

{Parrafo} {printf ("Se leyo parrafo: %s \n",yytext);}

/\*Distingue entre la ejecución interactiva y la ejecucion con archivo\*/

/\*Si se ejecuta con archivo indica el número de linea en el que se encuentra el error\*/

{ if (op==1)

{

printf("Linea: %d. Error lexico, no se reconoce: %s \n",linea,yytext);

}else{

printf("Error lexico, no se reconoce: %s \n",yytext);

}

}

**Sección de Código de Usuario:**

La sección de código de usuario es simplemente copiada a lex.yy.c. Esta sirve para incluir rutinas que llamarán o serán llamadas por el analizador. Incluimos esta parte en la 2da entrega.

%%

int main (int argc, char \*argv[])

{

/\* argc es un contador que tiene la cantidad de argumentos que le pasas. Siempre vale 1 o mas por que cuenta el nombre del ejecutable\*/

/\* por eso pregunto si es mayor a 1 que quiere decir que le pase un parametro (que seria el nombre del archivo de entrada)\*/

if (argc > 1)

{

/\*si es mayor a 1 quiere decir que le pase el txt de entrada con el codigo html\*/

/\* \*argv[] es un vector que contiene lo que se recibio por parametro, en la posicion 0 va a tener el nombre del ejecutable y en la 1 el nombre del archvo de entrada a analizar\*/

/\*fopen es para empezar a leer del archivo, recibe dos parametros, el primero es el nombre del archivo y el segundo es el modo que sería lectura (rt)\*/

linea=1;

op=1;

yyin = fopen(argv[1], "rt");

}else{

/\*si argc es < 1 quiere decir que no le pasamos ningun parametro por lo tanto va a leer de lo que ingresemos por teclado\*/

yyin = stdin;

}

yylex();

return 0;

}

**Modo de Ejecución del Intérprete**

La ejecución del intérprete se realizará a través de la consola de comandos de

Windows y a partir de un archivo. El archivo del lexer debe tener la extensión “.l”.

Instrucciones para ejecutar el analizador léxico:

-Paso 1: en la terminal de comandos cambiamos de directorio, por el de nuestro

lexer a ejecutar y el archivo.txt donde se aloja el codigo a analizar. Se recomienda que sea en la misma partición del disco que se instaló Flex y Bison.

....>\cd “directorio del archivo lexer.l”

-Paso 2: procedemos a compilar nuestro analizador: ...>\flex lexer.l

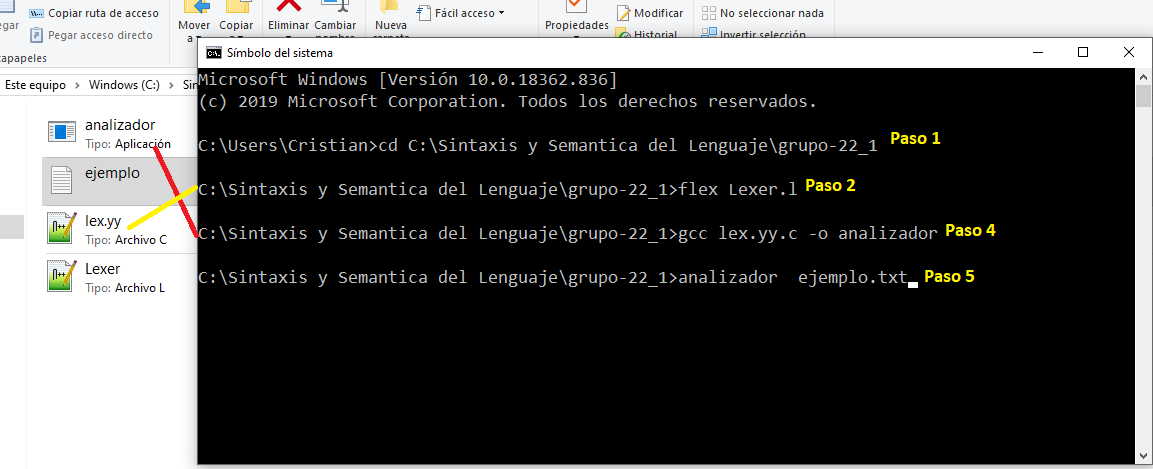
-Paso 3: en el paso anterior se generó un archivos llamado lex.yy.c :, el cual debemos compilar para generar el analizador ejecutable.

-Paso 4: Mediante la siguente sentencia generamos el ejecutable del analizador: …>\ gcc lex.yy.c -o analizador

-Paso 5: Por último le pasamos como parámetro nuestro archivo de entrada .txt con

el codigo a analizar: ...> analizador codigo.txt .

-Paso 6: Con lo anterior hecho, se visualizara la lista de tokens encontrados en el archivo .txt; para terminar su ejecución se debe presionar Ctrl+C o Ctrl+D.



**Ejemplo de ejecución del Lexer:**

Dado el siguiente ejemplo:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<title>HOLA MUNDO</title>

</head>

<body>

<SECTION id="principal">

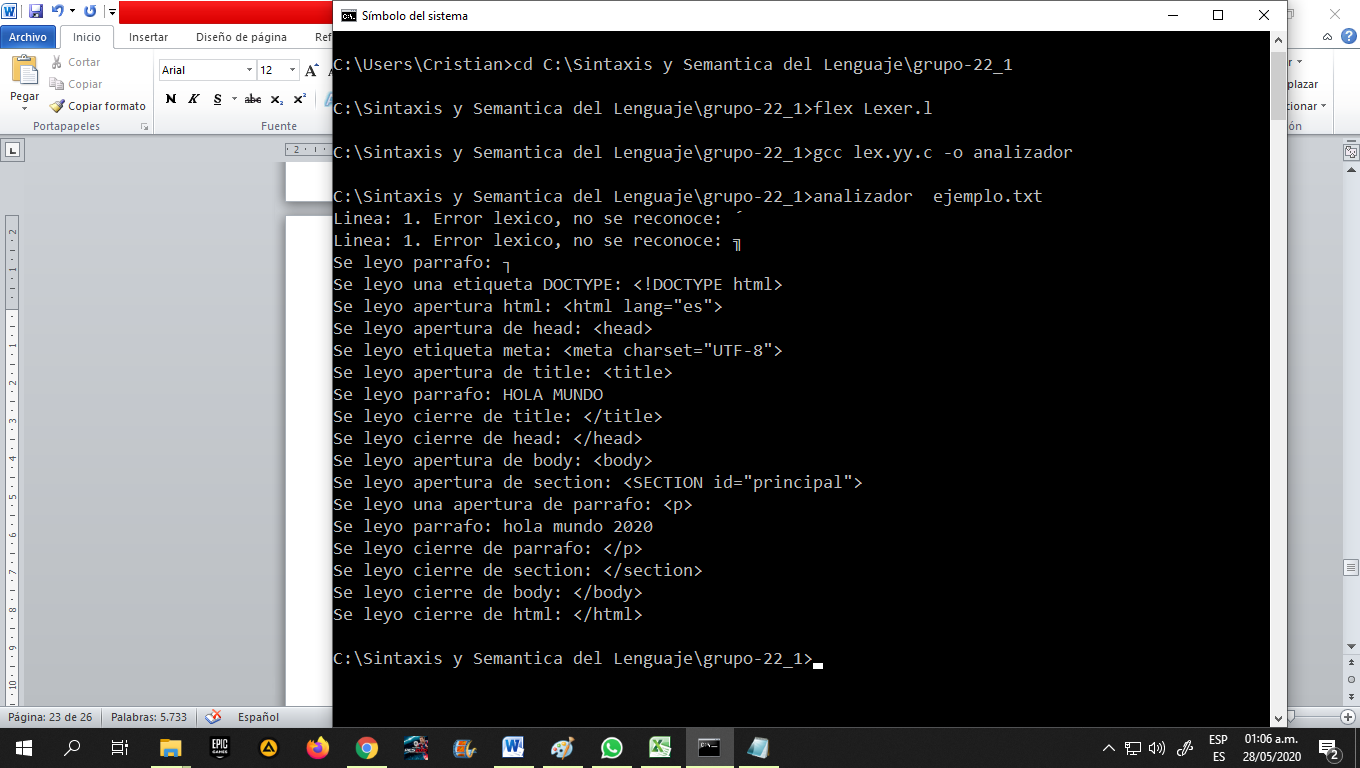
<p>hola mundo 2020</p>

</section>

</body>

</html>

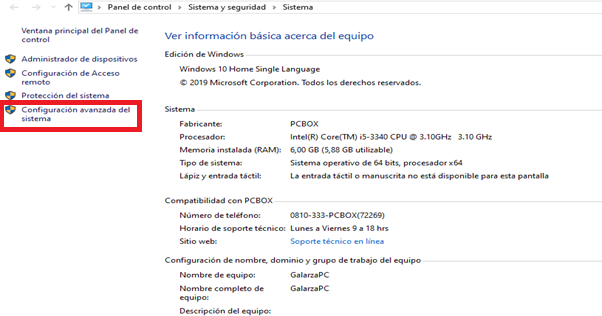
Al pasarlo al analizador lexer se visualizara lo siguiente:



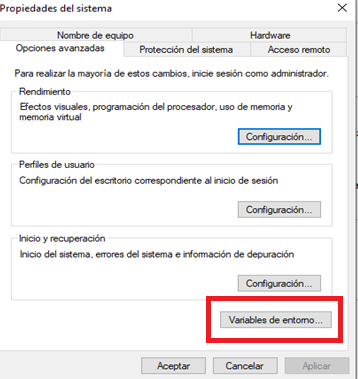
**Anexo**

**Instalación de los generadores Flex y Bison y su configuración para su posterior uso:**

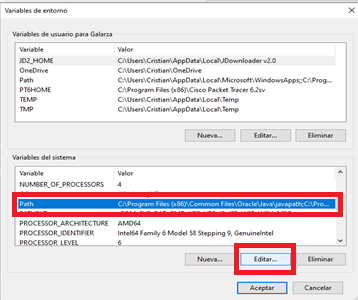
1. Descargamos el instalador de ambas herramientas de las siguientes dos páginas:
   1. Bison: https://sourceforge.net/projects/gnuwin32/files/bison/2.4.1/bison-2.4.1-setup.exe/download
   2. Flex: https://sourceforge.net/projects/gnuwin32/files/flex/2.5.4a-1/flex-2.5.4a-1.exe/download
2. Ambos se instalaran en una carpeta llamada GnuWin32, en el disco C por defecto.
3. Luego, accedemos al asistente “Propiedades del sistema”, estando ahí seleccionamos la opción “Configuración Avanzada del sistema ”



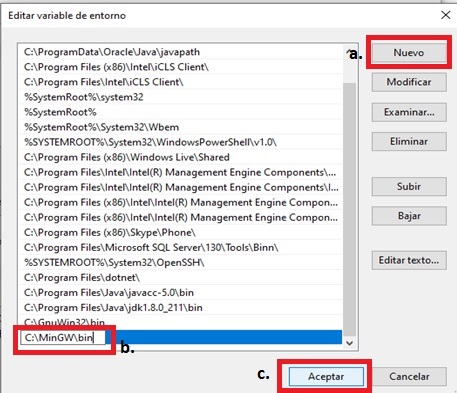
1. Un vez que la ventana se abre click en “Variables de entorno”



1. En el asistente, seleccionamos de “Variables de sistema” la variable ‘Path’ y click en editar:



1. Se abrira el asistente de “Editar Variable de Entorno”, procedemos de la siguiente manera:
   1. Seleccionamos “Nuevo”
   2. En la parte inferior se habilitara un espacio, en donde debemos escribir la url de la carpeta ‘bin’ donde están el Flex y Bison instalados. Que en este caso sería ‘C:\GnuWin32\bin’.
   3. Por ultimo solo debemos aceptar en los sucesivos asistentes abiertos.



1. Ahora ya se encuentra instalados tanto el Bison y Flex para manejar desde consola de comandos.

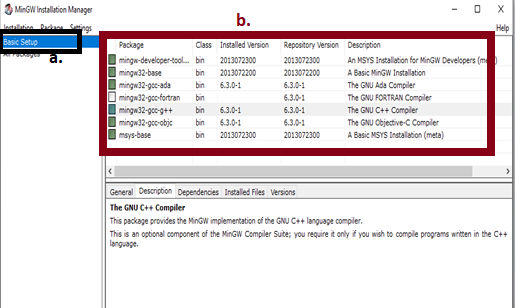
**Instalación del programa MinGW**

MinGW , una contracción de "Minimalist GNU for Windows", es un entorno de desarrollo minimalista para aplicaciones nativas de Microsoft Windows. Los compiladores de MinGW proporcionan acceso a la funcionalidad del tiempo de ejecución de Microsoft C y algunos tiempos de ejecución específicos del lenguaje.

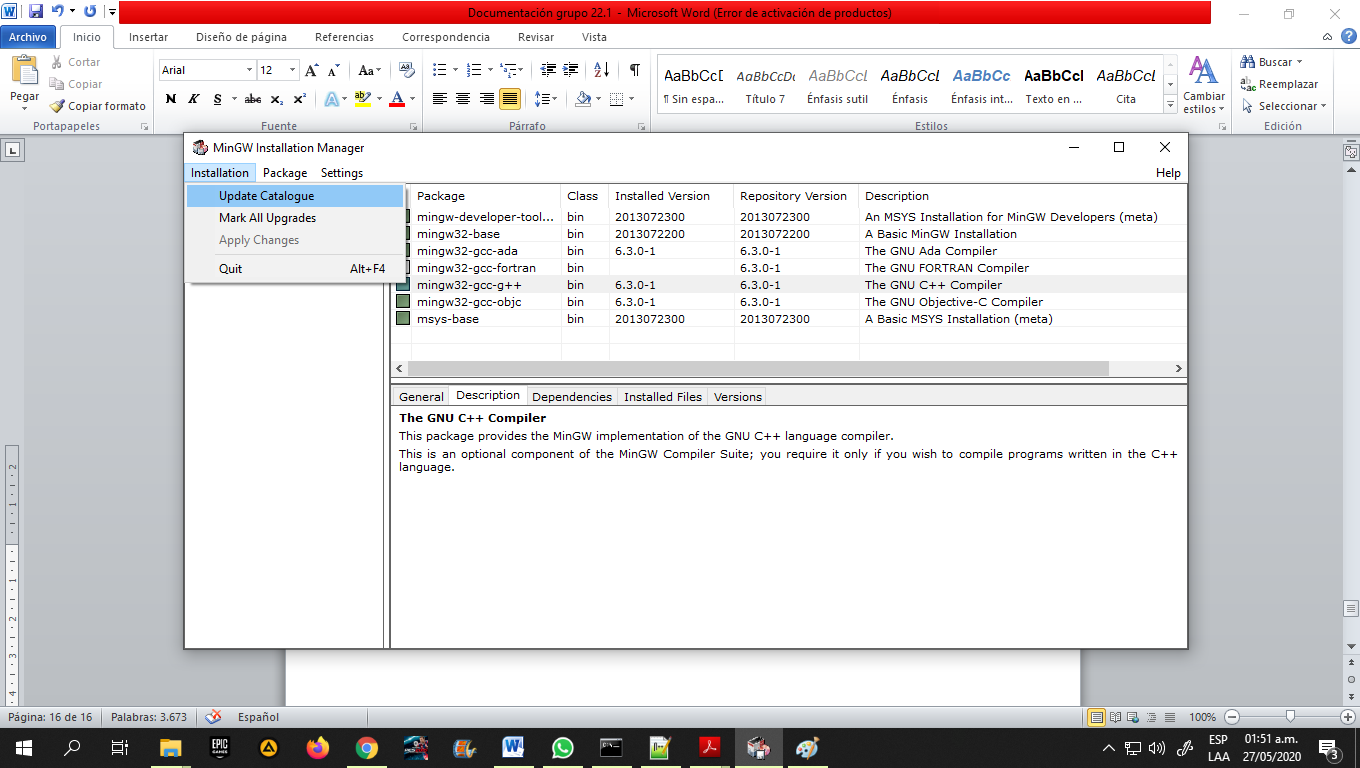
1. Descargar el programa desde el siguiente enlace, tanto para 32bits como 64 bits :

<https://sourceforge.net/projects/mingw-w64/>

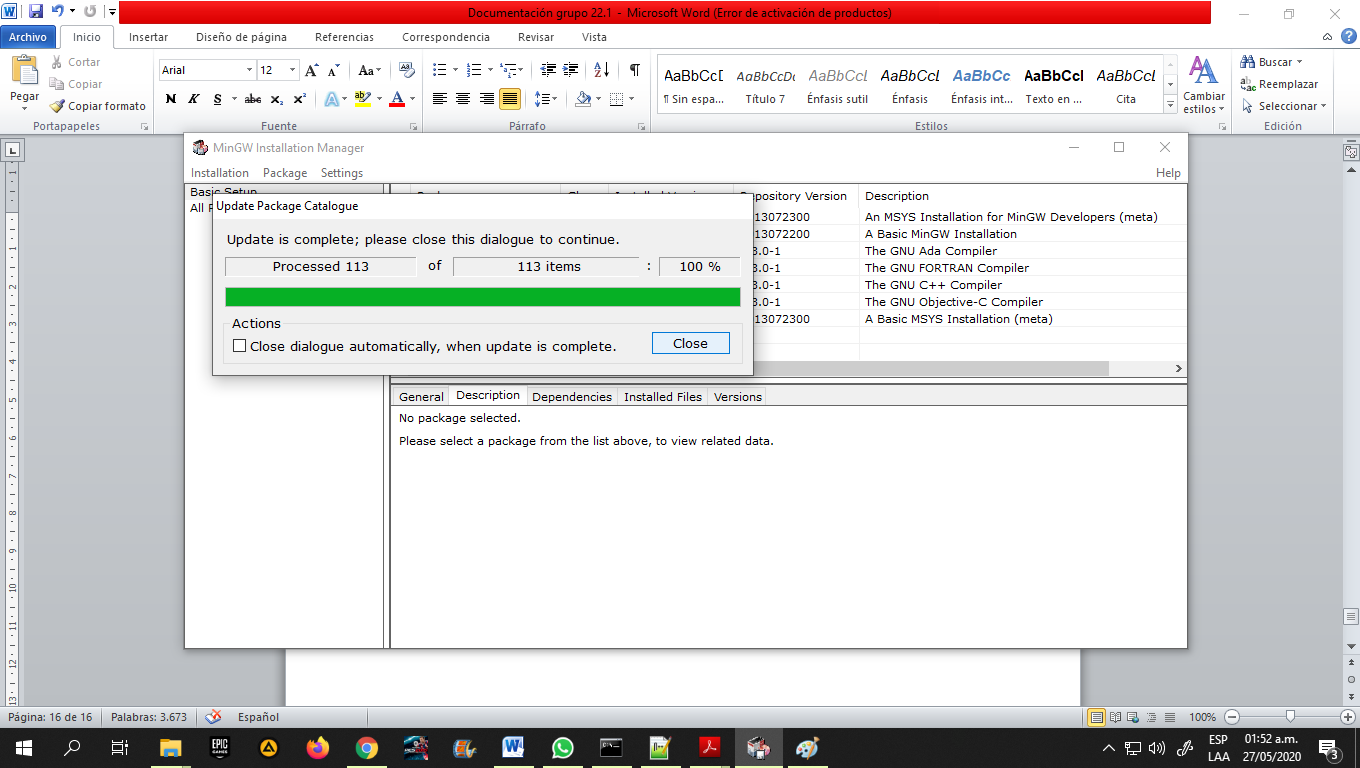
1. Luego abrimos la aplicación:
   1. Seleccionamos “Basic Setup” desde el menú de la izquierda;
   2. A la derecha seleccionamos todos menos
      1. mingw32-gcc-ada
      2. mingw32-gcc-fortran



* 1. De la barra de menú seleccionamos “Installation” , luego “Update Catalogue”



* 1. Nos aparecerá una ventana donde se muestra la actualización y al terminar la cerramos:



1. Por último la carpeta bin de MinGW, guardamos su url C:\MinGW\bin, en variables de entorno. Se procede de la misma manera que se realizó con Flex y Bison