U. t. n. – FACULTAD REGIONAL RESISTENCIA

Ingeniería en Sistemas de Información

Primer Cuatrimestre – 2020

SINTAXIS Y SEMANTICA DE LOS LENGUAJES

**Trabajo Práctico Integrador:** **Diseño e Implementación de Lexer y Parser**

InTEGRANTES:

* encinas, maría del alba
* llopi, ulises alan
* martínez cundom, andrea agustina
* stacul, nahiara fernanda

RESISTENCIA, Sabado 27 de Junio de 2020

SSL 2020

# **ÍNDICE**

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

**Lenguaje HTML**

Componentes Lexicos……………………………………… ………………...………..…...……3

Atributos globales y específicos…………………………………..…………...………….…...…4

Observaciones……….……………………………………… ………………...…………..…..…5

**Definición de la GRAMÁTICA**

Declaración de símbolos terminales y no terminales ….…………………………….....……...…7

Gramática libre de contexto………………………………………………………………......…..8

Reglas de producción de la gramática………………………………..………………………….12

**Análisis Léxico**

Sección de declaraciones…………………....…………...………………………....….………....15

Sección de reglas……………………………………… ………………...……………………....15

**Análisis Sintáctico**……………………………………………………………………………….25

Declaraciones de Bison……………………………………… ………………...……………......26

Declaraciones de las reglas gramaticales….………………… ………………...…………….....28

Código C………………………………...….………………… ………………...……...…….....36

Pasos para ejecutar el Parser……………………………………… ……………….....……..…..38

**Funciones Auxiliares**……………….…………………… ………………...………………..….40

**Modo de obtención y ejecución del intérprete** …………………………………………….….41

**Ejemplos**……………………………………… ………………...……...….................................43

**Conclusion**………………………………………… ………………...……...…..........................51

**Bibliografía**……………………...………………………………………….…………………....52

# **INTRODUCCIÓN**

En el presente trabajo para la asignatura de Sintaxis y Semántica de los Lenguajes realizamos como primera instancia del Trabajo Práctico Integrador la gramática para el lenguaje HTML5, para ello separamos el trabajo en tres partes: Definición de la Gramática a utilizar, Analizador Léxico y Analizador Sintáctico.

Finalmente con ayuda de Bison versión: 2.4.2 generamos los archivos archivos y.tab.h,y.tab a partir del ingreso en el comando de sistema de nuestro parser.y, en el código del mismo incluimos el orden correcto de cada sentencia en html, de este modo si se ingresara una estructura incorrecta pueda mostrarnos por pantalla un mensaje de error.

# **LENGUAJE HTML5**

***Componentes Léxicos***

* Etiquetas de HTML

Las etiquetas en HTML se definen con aperturas y cierres, las cuales se diferencias en que los cierres llevan una barra la cual los identifica, además cada etiqueta van encerrados entre brackets “<”, “>”

Algunas etiquetas especiales no llevan cierre tales como: br (salto de línea), hr (salto de línea con línea horizontal.

<!doctype html>

<html> </html>

<head> </head>;

<meta charset=”UTF-8”>

<meta >

<title> </title> ;

<body> </body>

<section> </section>

<div> </div>

<p> </p>

<h1> </h1>

<h2> </h2>

<h3> </h3>

<h4> </h4>

<h5> </h5>

<h6> </h6>

<ol> </ol>

<ul> </ul>

<li> </li>

<table> </table>

<thead> </thead>

<tfoot> </tfoot>

<tbody> </tbody>

<tr> </tr>

<td> </td>

<th> </th>

<a> </a>

<img

* Atributos

1. *Atributos globales:* son aquellos aplicables a todas las etiquetas que van dentro del cuerpo del html, ya sea en conjunto con los atributos específicos o solos.

class

id

1. *Atributos Específicos:* son aquellos que se corresponden con una etiqueta particular, y como se mencionó anteriormente pueden ir acompañados de los atributos globales.

* **meta**

name

content

* **Listas**

type (ol, ul)

value (li)

* **a**

href

target

* **img**

src

alt

width

height

* **tabla**

border

<caption> </caption>

***Sentencias***

La estructura Básica de Html viene dada de la siguiente manera:

El lenguaje contiene un bloque de Cabecera que se define con la etiqueta <head> , en este bloque va el título de la página, meta charset que sirve para que el código tenga compatibilidad con distintos tipos de pantalla como así también algunas actualizaciones del lenguaje. Luego está la estructura de Cuerpo definida como <body> donde se describen las acciones que queremos que nuestro programa realice, en él podemos incluir listas ordenadas o no ordenadas indicadas con las etiquetas <ul> u <ol> respectivamente, o incluir ítems dentro de las listas con la etiqueta <li> , también podemos agregar tablas <table> definiendo <thead> para las filas de cabecera, <tbody> para el cuerpo de tabla y <tfoot> para el pie de la misma, utilizamos la etiqueta <tr> para indicar las filas de la tabla y <th> para las celdas de cabecera, para las generales usamos <td>

<!Doctype html>

<html lang=”es”> /\* aca se indica el idioma de nuestro código

<head> /\* la cabecera del programa

< meta charset=”UTF-8“> /\* define la codificación de los caracteres

<body> /\*indica el comienzo del cuerpo de nuestro código

</body> /\*indica el cierre del cuerpo de nuestro código

</html> /\*

**Observaciones:** Se ha limitado el lenguaje del intérprete. El analizador reconocerá todo el código en HTML escrito menos algunas restricciones :

* Es posible meter una lista dentro de otra, incluso mezclando.
* Las etiquetas de header pueden ser <h1> y/o <h2>. Además solo pueden contener texto plano (no otras etiquetas).
* Las etiquetas de lista <ul> <ol> solo contendrán elementos de ítem de lista <li>, donde estas últimas pueden contener lo que sea.
* Las tablas, contendrán solo la estructura (filas, theader tfooter, tbody)
* Los elementos de celda <th> y <td> puedan tener lo que sea.
* La etiqueta <head> contendrá sólo las etiquetas meta y title
* Las etiquetas <title> contiene solo texto dentro
* Las etiquetas meta decimos que es obligatorio que venga “meta charset” (lo cual obliga que haya un head) y además pueden venir otras genéricas (con name y content).
* No nos importa que venga dentro de name ="xx" y dentro de content="xx" solo que sea texto.
* 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" .. )

Simplificando lo especificado anteriormente:

* Atributos Globales (para todos): CLASS y ID
* Atributos Específicos, según el elemento (tag):
  + <ol> TYPE
  + <li> VALUE
  + <a> HREF (requerido) y TARGET
  + <img> ALT SRC(requerido si o si) WITH HEIGHT:
  + <html> Lang (requerido)
  + <table> BORDER
  + <meta> dichas mas arriba
* RESTRICCIONES QUE HEMOS DEFINIDO PARA EL INTERPRETE:

El interprete no reconoce espacios dentro de las etiquetas, por ejemplo < p> o <p >, solo en la apertura de html y la etiqueta meta charset son permitidas al final.

El interprete no reconoce tildes cuando se realiza el analisis de un archivo, solo desde el modo interactivo

# **DEFINICIÓN DE LA GRAMÁTICA**

La gramática diseñada para este trabajo es Libre de Contexto. Para su elaboración, en primer lugar hemos establecido los símbolos terminales y no terminales utilizados, en segundo lugar la propia gramática para el lenguaje y finalmente una breve descripción sobre las reglas de la gramática, donde se encuentran una definición general sobre las etiquetas y los símbolos no terminales utilizados para cada caso.

* ***Declaración de símbolos terminales y no terminales***

|  |  |
| --- | --- |
| Terminales | <!doctype html ; <html lang =“ “>; </html> ; <head ; </head>;  <body ; </body> ; <meta charset=”UTF-8”> ; <meta, name=“ , content=” ;  <title *;* </title> ; <section ; </section> ; <div ; </div> ; <p ; </p> ; <h1 ; </h1> ; <h2 ; </h2> ; <h3 ; </h3> ; <h4 ;</h4> ; <h5 ; </h5> ; <h6 ; </h6> ; <ol ; </ol> ;<ul; </ul>; <li ; </li> ; <table ; </table> ; <thead ; </thead> ; <tfoot ; </tfoot> ; <tbody ; </tbody> ; <tr> ; </tr> ; <td> ; </td> ; <th> ; </th> ; <a> ; </a> ; <img; src= “ ; class=” ; id=” ; type=” ; value\_=” ; href=” ” ; target=” ” ; alt= “ ”; width=” ”; height=”” ; border= “ ” ; <caption> ; </caption> |
| No terminales | Z; S; HEAD; BODY; MN; MC; T; E; SECDIV; PARRAF; H; SALTO; TEXTO; OL; UL ; LI; TABLA; CUETAB; THEAD; TFOOT; TBODY; FILAS; CELDAS; URL; IMG; A; APC; API; AO, AL; AUH; AUT; AIMG; ATRIMAGEN; AT; ATC |

* ***Gramatica***

𝚺 → Z

**Z** → <!doctype html> **HTML**

**HTML** → <html>**HEAD BODY**</html>

**HEAD** → <head>**MC MN TITLE**</head> | <head>**MC MN** </head>

**MC** → <meta charset=”UTF-8”>

**MN** → <meta name=“**TXT** **CIERRECOMILLAS** content=”**TXT** **CIERRECOMILLAS** **CIERRETAG MN** | <meta name=“**TXT** **CIERRECOMILLAS** **CIERRETAG MN** |

<meta content=”**TXT** **CIERRECOMILLAS** **CIERRETAG MN** | <meta name=“**TXT** **CIERRECOMILLAS** content=”**TXT** **CIERRECOMILLAS** **CIERRETAG** | <meta name=“**TXT** **CIERRECOMILLAS** **CIERRETAG** | <meta content=”**TXT** **CIERRECOMILLAS** **CIERRETAG |**

**T**→ <title>**TXT**</title>

**BODY**→ <body **CIERRETAG** **E**</body>

**E** → SEC | DIV | PARRAFO | H | SALTO | TEXTO | OL | UL | LI | TABLA | ELA | IMG | AN |SEC E | DIV E | PARRAFO E | H E | SALTO E| TEXTO E | OL E | UL E | LI E | TABLA E | URL E | IMG E | AN E

**SEC**→ <section **A CIERRETAG** **E** </section> | <section **CIERRETAG E** </section>

**DIV→** <div **A CIERRETAG** **E** </div> | <div**CIERRETAG** **E** </div>

**PARRAFO** → <p **A CIERRETAG E**</p> | <p **CIERRETAG E**</p>

**H** → <h1 **A** **CIERRETAG TXT** </h1> | <h2 **A** **CIERRETAG TXT**</h2> | <h3 **A** **CIERRETAG TXT** </h3> | <h4 **A** **CIERRETAG TXT** </h4> | <h5**A** **CIERRETAG TXT** </h5> | <h6**A** **CIERRETAG** **TXT** </h6> | <h1 **CIERRETAG** **TXT**</h1> | <h2 **CIERRETAG** **AN** </h2> | <h3 **CIERRETAG** **TXT**</h3> | <h4 **CIERRETAG** **TXT**</h4> | <h5 **CIERRETAGTXT** </h5> | <h6 **CIERRETAG TXT** </h6>

**SALTO** → <br **A** **CIERRETAG** | <br **CIERRETAG** | <hr **A** **CIERRETAG** | <hr **CIERRETAG**

**TEXTO** → <strong **CIERRETAG** **E**</strong> | <em **CIERRETAG E**</em> | <mark **CIERRETAG E**</mark>

**OL** → <ol **AO** **CIERRETAG** **LI** </ol> | <ol **CIERRETAG**  **LI** </ol>

**UL** → <ul **AO** **CIERRETAG** **LI** </ul> | <ul **CIERRETAG** **LI** </ul>

**LI**→ <li **AL CIERRETAG** **E** </li> **LI** | <li AL **CIERRETAG** **E LI** </li> **LI** | <li AL **CIERRETAG**  **LI** </li> **LI** | <li AL **CIERRETAG** **E** </li> | <li AL **CIERRETAG**  **LI** </li> | <li **CIERRETAG** **E** </li> **LI** | <li **CIERRETAG** **E** **LI** </li> **LI |** <li **CIERRETAG** **LI** </li> **LI** | <li **CIERRETAG** **E** </li> **LI** | <li **CIERRETAG** **LI** </li>

**TABLA** → <table **A AT** **CIERRETAG ATC FILAS CUETAB** </table>

| <table **A AT CIERRETAG**  **ATC CUETAB** </table>

|<table **A AT** **CIERRETAG** **ATC FILAS** </table>

| <table **A AT** **CIERRETAG** **FILAS CUETAB**</table>

| <table **A AT** **CIERRETAG**  **FILAS**</table>

| <table **A AT** **CIERRETAG**  **CUETAB** </table>

| <table **AT A** **CIERRETAG ATC FILAS CUETAB** </table>

| <table **AT A** **CIERRETAG**  **ATC CUETAB** </table>

|<table **AT A** **CIERRETAG**  **ATC FILAS** </table>

| <table **AT A** **CIERRETAG** **FILAS CUETAB**</table>

| <table **AT A** **CIERRETAG** **FILAS**</table>

| <table **AT A** **CIERRETAG**  **CUETAB** </table>

| <table **A** **CIERRETAG** **ATC FILAS CUETAB** </table>

| <table **A** **CIERRETAG**  **ATC CUETAB** </table>

| <table **A** **CIERRETAG**  **ATC FILAS**</table>

| <table **A** **CIERRETAG** **FILAS CUETAB** </table>

| <table **A** **CIERRETAG** **CUETAB** </table>

| <table **A** **CIERRETAG** **FILAS** </table>

| <table **AT** **CIERRETAG**  **ATC FILAS CUETAB** </table>

| <table **AT** **CIERRETAG** **ATC CUETAB** </table>

| <table **AT** **CIERRETAG**  **ATC FILAS**</table>

| <table **AT CIERRETAG** **FILAS CUETAB** </table>

| <table **AT** **CIERRETAG** **CUETAB** </table>

| <table **AT** **CIERRETAG** **FILAS** </table>

|<table **CIERRETAG**  **ATC FILAS CUETAB** </table>

|<table **CIERRETAG**  **ATC CUETAB** </table>

|<table **CIERRETAG**  **ATC FILAS** </table>

|<table **CIERRETAG** **FILAS CUETAB** </table>

|<table **CIERRETAG** **FILAS** </table>

|<table **CIERRETAG**  **CUETAB** </table>

**CUETAB** → **TH TB TF | TH TF TB | TB TH TF | TB TF TH |TF TB TH | TF TH TB | TH TB |TH TF | TB TH | TB TF | TF TB | TF TH | TH | TB | TF |** **TH TB TF FILAS | TH TF TB FILAS | TB TH TF FILAS | TB TF TH FILAS |TF TB TH FILAS | TF TH TB FILAS | TH TB FILAS |TH TF FILAS | TB TH FILAS | TB TF FILAS | TF TB FILAS | TF TH FILAS | TH FILAS | TB FILAS | TF FILAS**

**TH** → <thead **CIERRETAG** **FILAS** </thead>

**TF** → <tfoot **CIERRETAG** **FILAS** </tfoot>

**TB**→ <tbody **CIERRETAG**  **FILAS** </tbody>

**FILAS** → <tr **CIERRETAG**  **TD** </tr> | <tr **CIERRETAG**  **TD** </tr>**FILAS** | <tr **CIERRETAG**  **CELDAS** </tr> | <tr **CIERRETAG**  **CELDAS** </tr>**FILAS** | <tr **CIERRETAG** </tr> | <tr **CIERRETAG** </tr>**FILAS**

**TD**→ <td **CIERRETAG**  **E** </td> **TD** | <td **CIERRETAG**  **E** </td> **CELDAS** | <td **CIERRETAG**  **E** </tr> | <td **CIERRETAG** </td>**FILAS** | <td **CIERRETAG** </td> | <td **CIERRETAG** </td>**FILAS**

**CELDAS**→ <th **CIERRETAG** **E** </th> **TD** | <th **CIERRETAG** **E** </th> **CELDAS** | <th **CIERRETAG** **E**</th> | <th **CIERRETAG** </th> **CELDAS** | <th **CIERRETAG** </th>

**ATC**→ <caption  **CIERRETAG TEXTO** </caption>

**ELA →** <a **A AUH AUT CIERRETAG E** </a>

|<a **AUH A AUT CIERRETAG E** </a>

|<a **AUH AUT A CIERRETAG E** </a>

|<a **A AUH CIERRETAG E** </a>

|<a **AUH A CIERRETAG E** </a>

|<a **AUH AUT CIERRETAG E** </a>

|<a **AUT AUH CIERRETAG E** </a>

|<a **AUH CIERRETAG E** </a>

**IMG**→ <img **SRC A AIMG ATRIMAGENWH CIERRETAG**

|<img **A SRC AIMG ATRIMAGENWH CIERRETAG**

|<img **A AIMG SRC ATRIMAGENWH CIERRETAG**

|<img **A AIMG ATRIMAGENWH SRC CIERRETAG**

|<img **SRC ATRIMAGENWH A AIMG CIERRETAG**

|<img **ATRIMAGENWH SRC A AIMG CIERRETAG**

|<img **ATRIMAGENWH A SRC AIMG CIERRETAG**

|<img **ATRIMAGENWH A AIMG SRC CIERRETAG**

|<img **SRC ATRIMAGENWH AIMG A CIERRETAG**

|<img **ATRIMAGENWH SRC AIMG A CIERRETAG**

|<img **ATRIMAGENWH AIMG SRC A CIERRETAG**

|<img **ATRIMAGENWH AIMG** **SRC CIERRETAG**

|<img **SRC AIMG ATRIMAGENWH CIERRETAG**

|<img **AIMG SRC ATRIMAGENWH CIERRETAG**

|<img **AIMG ATRIMAGENWH SRC CIERRETAG**

|<img **SRC ATRIMAGENWH AIMG CIERRETAG**

|<img **ATRIMAGENWH SRC AIMG CIERRETAG**

|<img **ATRIMAGENWH AIMG SRC CIERRETAG**

|<img **SRC AIMG CIERRETAG**

|<img **AIMG SRC CIERRETAG**

|<img **SRC ATRIMAGENWH CIERRETAG**

|<img **ATRIMAGENWH SRC CIERRETAG**

|<img **SRC A CIERRETAG**

|<img **A SRC CIERRETAG**

|<img **SRC CIERRETAG**

**A** → **APC** **API | API APC| APC | API**

**APC**→ class=”**AN**”

**API**→ id=”**AN**”

**AO**→ type=”**AN”**

**AL**→ value\_=”**AN**”

**AUH**→ href=”**URL**”

**AUT**→ target=”\_blank | \_parent | \\_self | \\_top”

**SRC→** src=”**URL**”

**AIMG** → alt= “**AN**”

**ATRIMAGENWH**→ width=”**NUM** **CIERRECOMILLAS** height= ”**NUM** **CIERRECOMILLAS**

|height= ”**NUM** **CIERRECOMILLAS** width=”**NUM** **CIERRECOMILLAS**

**|**width=”**NUM** **CIERRECOMILLAS**

|height= ”**NUM** **CIERRECOMILLAS**

**AT**→ border= “**AN**”

**CIERRETAG→ >**

**CIERRECOMILLAS →** “

**TXT** → **AN TXT** | **AN**

**TYN→ AN TXT | NUM TXT | AN | NUM**

**Aclaraciónes:**

Para el orden de las producciones, tuvimos en cuenta el orden de los tokens y etiquetas del pdf en el que se encuentran las especificaciones dadas para el TFI, brindado por la cátedra.

Se utiliza “ **|** ” para indicar la separación entre producciones que corresponden a un mismo símbolo no terminal.

* **Reglas de producción de la gramática**

**No Terminales utilizados para la Estructura del Documento:**

Z; S; HEAD; BODY; MN; MC; T

**No terminal que contiene a las etiquetas utilizadas en el cuerpo:**

E

**Etiquetas básicas de párrafo:**

SEC;DIV; PARRAFO, H, SALTO; TEXTO

**Listas:**

OL; UL ; LI

**Tablas:**

TABLA; CUETAB; THEAD; TFOOT; TBODY; FILAS; CELDAS

**Enlaces:**

URL

**Imagenes:**

IMG

**Atributos Globales :**

A; APC; API

**Atributos Específicos:**

**Lista:**

AO; AL

**Elemento <a>:**

AUH; AUT

**Imagenes:**

AIMG; ATRIMAGENWH

**Tablas:**

AT; ATC

**Texto y números:**

TXT; TYN

**Tokens:**

URL; TOKEN\_TARG; AN; CIERRETAG, CIERRECOMILLAS

# **ANÁLISIS LÉXICO**

El archivo “Lexer-G22.2.l” está compuesto inicialmente por las expresiones regulares del lenguaje HTML5, seguidas por sus reglas gramaticales. El mismo fue cargado en el programa Flex que genera un archivo “yy” reconocido y ejecutable en lenguaje C, conocido como Lexer. Éste, identifica las etiquetas o tokens que pertenecen al lenguaje html imprimiendo por pantalla error en caso de encontrarse con palabras que no pertenecen ni son reconocidas y en caso contrario indica si se trata de una etiqueta, atributo o token.

Elegimos el lenguaje C por nuestras preferencias y actuales conocimientos. Lo desarrollamos en el editor de texto "Sublime\_Text".

**Aclaramos que para poder ejecutar y compilar el analizador léxico se requiere tener descargado Flex, la versión que utilizamos fue: 2.5.35**

*“Flex es un una herramienta que permite generar analizadores léxicos. A partir de un conjunto de expresiones regulares, Flex busca concordancias en un fichero de entrada y ejecuta acciones asociadas a estas expresiones.”*

A la hora de ejecutar el archivo desde los comandos del sistema utilizamos la extensión “-i” lo que nos permite que el análisis sea case -insensitive, es decir que es indiferente a mayusculas o minusculas, siendo correcto ingresar por ejemplo <h> o <H>.

**·*Sección de Declaraciones***

|  |
| --- |
| **html (<html[\ ]lang=(\"([a-z]{2})\")[\ ]?>)**  **metanc (<meta)**  **name (name=[\ ]?\")**  **content (content=[\ ]?\")**  **atgc (class=[\ ]?\"[\ ]?[a-z\ A-Z0-9]\*[\ ]?\")**  **atgid (id=[\ ]?\"[\ ]?[a-z\ A-Z0-9]\*[\ ]?\")**  **atype (type=[\ ]?\"[a-zA-Z0-9]\*\")**  **atvalue (value=([\ ]?\"[0-9]\*\"))**  **athref (href=[\ ]?\"((http(s)?|ftp(s)?):\/\/(w{3}.)?(([0-9]\*[a-zA-Z]\*[0-9]\*.)\*[a-zA-Z]{2,3})?(\:[0-9]+)?(\/[a-zA-Z0-9]\*)?((.[a-zA-Z]+)?)\*)?(\#[a-zA-Z0-9]+)?[\ ]?\")**  **atarget (target=[\ ]?\"\\_blank\"|\"\\_parent\"|\"\\_self\"|\"\\_top\")**  **atalt (alt=[\ ]?\"[\ ]?[a-z\ A-Z0-9]\*[\ ]?\")**  **atw (width=[\ ]?\")**  **ath (height=[\ ]?\")**  **atb (border=[\ ]?\"[0-9]+[\ ]?\")**  **atc (<caption>)**  **cierreatc (<\/caption>)**  **atsrc (src=([\ ]?\"((http(s)?|ftp(s)?):\/\/(w{3}.)?)?(([0-9]\*[a-zA-Z]\*[0-9]\*.)\*[a-zA-Z]{2,3})?(\:[0-9]+)?(\/[a-zA-Z0-9]\*(.[a-zA-Z]+)?)\*(\#[a-zA-Z0-9]+)?[\ ]?\"))**  **metacharset (<meta[\ ]charset=(\"(UTF-8)\")[\ ]?>)**  **header (<h[1-6]{1})**  **cheader (<\/h[1-6]{1}>)**  **cierrecom [\"]**  **an [a-zA-Z]+|\=|\"|[\?]|\¿|\!|\¡|\.|\,|\:|\;|[\\*]|\/|\(|\)|\[|\]|\{|\}|\+|\\_|\-|\^|\#|\%|\&|\-**  **num [0-9]+(\.[0-9]+)?**  **url ((http(s)?|ftp(s)?):\/\/(w{3}.)?(([0-9]\*[a-zA-Z]\*[0-9]\*.)\*[a-zA-Z]{2,3})?(\:[0-9]+)?(\/[a-zA-Z0-9]\*(.[a-zA-Z]+)?)\*(\#[a-zA-Z0-9]+)?)**  **spc [\ ]\***  **tab [\t]\***  **ESP [\n]** |

**·*Sección deReglas***

|  |
| --- |
| **"<!doctype html>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(DOCTYPE);}**  **"<html>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(AHTML);}**  **"</html>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CHTML);}**  **"<head>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(AHEAD);}**  **"</head>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CHEAD);}**  **"<body>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ABODY);}**  **"</body>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CBODY);}**  **"<title>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ATITLE);}**  **"</title>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **{if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ASECTION);}**  **"</section>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CSECTION);}**  **"<div" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ADIV);}**  **"</div>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CDIV);}**  **"<p" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(APARRAFO);}**  **"</p>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CPARRAFO);}**  **"<br" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(BR);}**  **"<hr" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(HR);}**  **"<strong" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ASTRONG);}**  **"</strong>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CSTRONG);}**  **"<em" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(AEM);}**  **"</em>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CEM);}**  **"<mark" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(AMARK);}**  **"</mark>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CMARK);}**  **"<ol" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(AOL);}**  **"</ol>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(COL);}**  **"<ul" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(AUL);}**  **"</ul>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CUL);}**  **"<li" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ALI);}**  **"</li>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CLI);}**  **"<table" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ATABLE);}**  **"</table>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CTABLE);}**  **"<thead" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ATHEAD);}**  **"</thead>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CTHEAD);}**  **"<tbody" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ATBODY);}**  **"</tbody>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CTBODY);}**  **"<tfoot" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(AFOOT);}**  **"</tfoot>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CFOOT);}**  **"<tr" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ATR);}**  **"</tr>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CTR);}**  **"<td" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ATD);}**  **"</td>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CTD);}**  **"<th" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ATH);}**  **"</th>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CTH);}**  **"<a" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(AA);}**  **"</a>" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CA);}**  **"<img" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(IMG);}**  **">" {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CBRACKET);}**  **{html} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(AHTML);}**  **{metacharset} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(METACHARSET);}**  **{metanc} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(META);}**  **{name} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(NAME);}**  **{content} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CONTENT);}**  **{header} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(AHEADER);}**  **{cheader} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CHEADER);}**  **{atgc} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CLASS);}**  **{atgid} {**  **{if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ID);**  **}**  **{atype} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(TYPE);}**  **{atvalue} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(VALUE );}**  **{athref} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(HREF);}**  **{atarget} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(TARGET);}**  **{atalt} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ALT);}**  **{atw} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(WIDTH);}**  **{ath} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(HEIGHT);}**  **{atb} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(BORDER);}**  **{atc} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(ACAPTION);}**  **{cierreatc} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CCAPTION);}**  **{atsrc} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **}return(SRC);}**  **{cierrecom} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(CCOM);}**  **{url} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(URL);}**  **{an} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(TEXTO);}**  **{num} {if (yyin != stdin)**  **{**  **printf("%s",yytext);**  **}**  **return(NUM);}**  **{spc} {printf ("\ ");}**  **{tab} {printf ("\t");}**  **{ESP} {linea++;**  **printf ("\n");}** |

# **ANÁLISI SINTÁCTICO**

Utilizamos como compilador del analizador Sintáctico al Programa Bison que a su vez también es compatible con yacc de Unix.

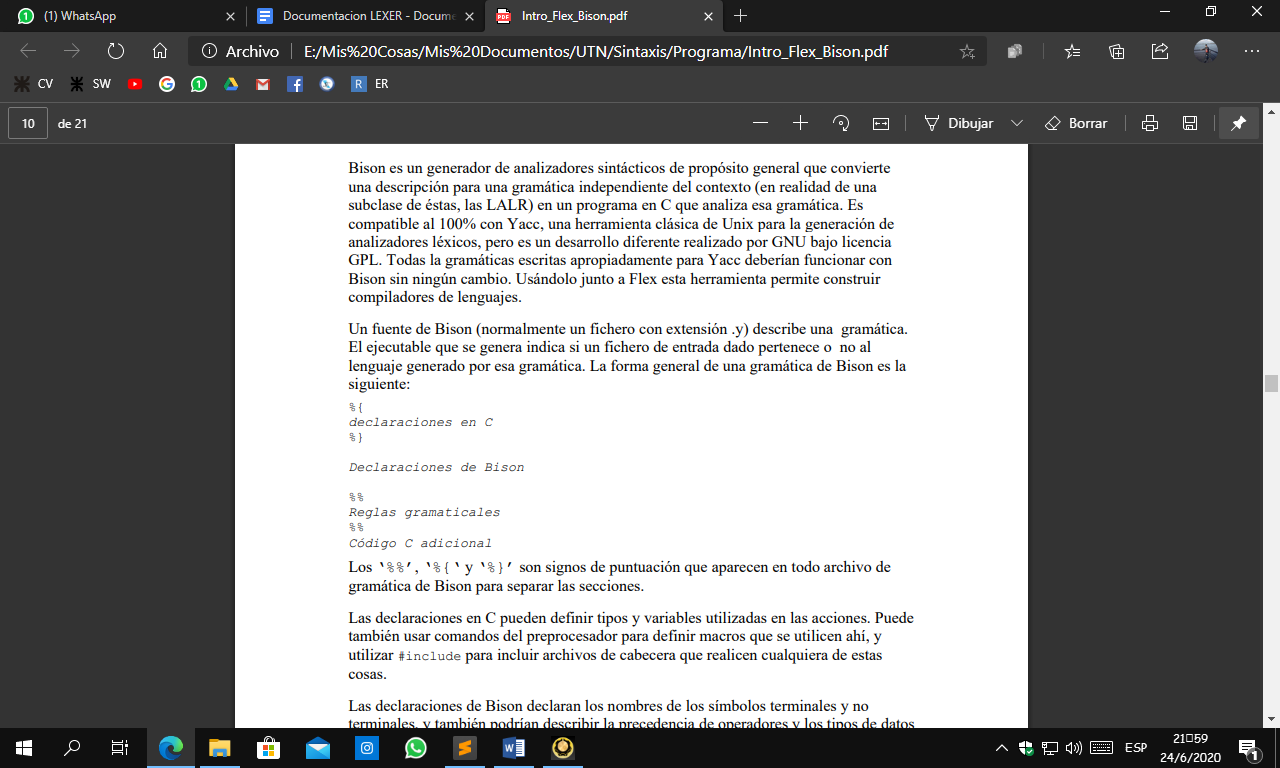
Bison toma las Gramáticas Libre de Contexto y con ello genera un programa C que con el mismísimo Parser realizado anteriormente llamado en este caso como Parser-G22.2.y

El programa Bison en su estructura consta de 4 bloques, primeramente con las Declaraciones en C, seguida por las Declaraciones de Bison, como tercer parte a las Reglas Gramaticales y por último parte para codigo en c adicional. En las declaraciones en C es el bloque donde estan las librerias y variables utilizadas en el programa.

Las declaraciones de Bison cuentan con los Tokens que son los No terminales utilizados en la gramática

La parte de las Reglas Gramaticales están ubicadas toda la Gramática Libre de Contexto y por último en el bloque de Código Adicional en C va un subprograma si necesitamos que realice una acción específica

La estructura debe estar definida de la siguiente manera:



**·*Declaraciones de Bison***

%start inicio

%token DOCTYPE

%token AHTML CHTML

%token AHEAD CHEAD

%token METACHARSET

%token META NAME CONTENT

%token ATITLE CTITLE

%token ABODY CBODY

%token ASECTION CSECTION

%token ADIV CDIV

%token APARRAFO CPARRAFO

%token AHEADER CHEADER

%token BR HR

%token ASTRONG CSTRONG

%token AEM CEM

%token AMARK CMARK

%token AOL COL AUL CUL ALI CLI

%token ATABLE CTABLE

%token ATHEAD CTHEAD

%token ATBODY CTBODY

%token AFOOT CFOOT

%token ATR CTR

%token ATD CTD

%token ATH CTH

%token AA CA

%token IMG

%token CBRACKET

%token CLASS ID

%token TYPE VALUE

%token HREF TARGET

%token ALT WIDTH HEIGHT SRC

%token BORDER ACAPTION CCAPTION

%token CCOM

%token URL

%token TEXTO

%token NUM

%token ESP

**·*Reglas Gramaticales***

inicio: DOCTYPE html5

html5: AHTML enc cuerpo CHTML

enc: AHEAD METACHARSET meta ATITLE text CTITLE CHEAD

|AHEAD meta METACHARSET ATITLE text CTITLE CHEAD

|AHEAD meta METACHARSET ATITLE CTITLE CHEAD

|AHEAD METACHARSET ATITLE text CTITLE CHEAD

|AHEAD METACHARSET ATITLE CTITLE CHEAD

;

meta: META NAME text CCOM CONTENT text CCOM CBRACKET meta

|META NAME text CCOM CBRACKET meta

|META CONTENT text CCOM CBRACKET meta

|META NAME text CCOM CONTENT text CCOM CBRACKET

|META NAME text CCOM CBRACKET

|META CONTENT text CCOM CBRACKET

text: TEXTO text | TEXTO

;

tyn: TEXTO tyn | TEXTO | NUM

;

cuerpo: ABODY etiqueta CBODY

;

atrglobal: CLASS ID

|CLASS

|ID

|ID CLASS

;

etiqueta: sec | div | par | h | salto | strong | mark | em | ol | ul | tabla | eta | imagen | TEXTO | NUM | URL | CCOM | sec etiqueta | div etiqueta | par etiqueta | h etiqueta | salto etiqueta | strong etiqueta | mark etiqueta | em etiqueta | ol etiqueta | ul etiqueta | tabla etiqueta | eta etiqueta | imagen etiqueta | TEXTO etiqueta | NUM etiqueta | URL etiqueta | CCOM etiqueta

;

sec: ASECTION atrglobal CBRACKET etiqueta CSECTION | ASECTION CBRACKET etiqueta CSECTION

;

div: ADIV atrglobal CBRACKET etiqueta CDIV | ADIV CBRACKET etiqueta CDIV

;

par: APARRAFO atrglobal CBRACKET etiqueta CPARRAFO | APARRAFO CBRACKET etiqueta CPARRAFO

;

h: AHEADER atrglobal CBRACKET tyn CHEADER | AHEADER CBRACKET tyn CHEADER

;

salto: BR atrglobal CBRACKET | BR CBRACKET | HR atrglobal CBRACKET | HR CBRACKET

;

strong: ASTRONG atrglobal CBRACKET etiqueta CSTRONG | ASTRONG CBRACKET etiqueta CSTRONG

;

mark : AMARK atrglobal CBRACKET etiqueta CMARK | AMARK CBRACKET etiqueta CMARK

;

em : AEM atrglobal CBRACKET etiqueta CEM | AEM CBRACKET etiqueta CEM

;

ol: AOL TYPE CBRACKET li COL | AOL CBRACKET li COL

;

ul: AUL TYPE CBRACKET li CUL | AUL CBRACKET li CUL

;

li : ALI VALUE CBRACKET etiqueta CLI li | ALI CBRACKET etiqueta CLI li | ALI VALUE CBRACKET etiqueta li CLI li | ALI CBRACKET etiqueta li CLI li | ALI VALUE CBRACKET li CLI li | ALI CBRACKET li CLI li | ALI VALUE CBRACKET etiqueta CLI | ALI CBRACKET etiqueta CLI

;

cuerpotabla: th tb tf

| th tf tb

| tb th tf

| tb tf th

| tf tb th

| tf th tb

| th tb

| th tf

| tb th

| tb tf

| tf tb

| tf th

| th

| tb

| tf

| th filas

| tb filas

| tf filas

| th tb filas

| th tf filas

| tb th filas

| tb tf filas

| tf tb filas

| tf th filas

| th tb tf filas

| th tf tb filas

| tb th tf filas

| tb tf th filas

| tf tb th filas

| tf th tb filas

;

th: ATHEAD CBRACKET filas CTHEAD

;

tb: ATBODY CBRACKET filas CTBODY

;

tf: AFOOT CBRACKET filas CFOOT

;

filas: ATR CBRACKET td CTR | ATR CBRACKET td CTR filas | ATR CBRACKET celdas CTR | ATR CBRACKET celdas CTR filas |ATR CBRACKET CTR filas |ATR CBRACKET CTR

;

td: ATD CBRACKET etiqueta CTD td | ATD CBRACKET etiqueta CTD celdas|ATD CBRACKET etiqueta CTD |ATD CBRACKET CTD td | ATD CBRACKET CTD celdas | ATD CBRACKET CTD

;

celdas:ATH CBRACKET etiqueta CTH td | ATH CBRACKET etiqueta CTH celdas | ATH CBRACKET etiqueta CTH |ATH CBRACKET CTH celdas | ATH CBRACKET CTH

;

caption: ACAPTION text CCAPTION

;

tabla: ATABLE atrglobal BORDER CBRACKET caption filas cuerpotabla CTABLE

|ATABLE atrglobal BORDER CBRACKET filas cuerpotabla CTABLE

|ATABLE atrglobal BORDER CBRACKET caption cuerpotabla CTABLE

|ATABLE atrglobal BORDER CBRACKET caption filas CTABLE

|ATABLE atrglobal BORDER CBRACKET filas CTABLE

|ATABLE atrglobal BORDER CBRACKET cuerpotabla CTABLE

|ATABLE BORDER atrglobal CBRACKET caption filas cuerpotabla CTABLE

|ATABLE BORDER atrglobal CBRACKET filas cuerpotabla CTABLE

|ATABLE BORDER atrglobal CBRACKET caption cuerpotabla CTABLE

|ATABLE BORDER atrglobal CBRACKET caption filas CTABLE

|ATABLE BORDER atrglobal CBRACKET filas CTABLE

|ATABLE BORDER atrglobal CBRACKET cuerpotabla CTABLE

|ATABLE atrglobal CBRACKET caption filas cuerpotabla CTABLE

|ATABLE atrglobal CBRACKET filas cuerpotabla CTABLE

|ATABLE atrglobal CBRACKET caption cuerpotabla CTABLE

|ATABLE atrglobal CBRACKET caption filas CTABLE

|ATABLE atrglobal CBRACKET cuerpotabla CTABLE

|ATABLE atrglobal CBRACKET filas CTABLE

|ATABLE BORDER CBRACKET filas cuerpotabla CTABLE

|ATABLE BORDER CBRACKET caption cuerpotabla CTABLE

|ATABLE BORDER CBRACKET caption filas CTABLE

|ATABLE BORDER CBRACKET filas CTABLE

|ATABLE BORDER CBRACKET cuerpotabla CTABLE

|ATABLE CBRACKET caption filas cuerpotabla CTABLE

|ATABLE CBRACKET caption cuerpotabla CTABLE

|ATABLE CBRACKET caption filas CTABLE

|ATABLE CBRACKET cuerpotabla CTABLE

|ATABLE CBRACKET filas cuerpotabla CTABLE

|ATABLE CBRACKET filas CTABLE

;

eta: AA atrglobal HREF TARGET CBRACKET etiqueta CA

|AA HREF atrglobal TARGET CBRACKET etiqueta CA

|AA HREF TARGET atrglobal CBRACKET etiqueta CA

|AA atrglobal HREF CBRACKET etiqueta CA

|AA HREF atrglobal CBRACKET etiqueta CA

|AA HREF TARGET CBRACKET etiqueta CA

|AA TARGET HREF CBRACKET etiqueta CA

|AA HREF CBRACKET etiqueta CA

;

atrwh: WIDTH NUM CCOM HEIGHT NUM CCOM

|WIDTH NUM CCOM

|HEIGHT NUM CCOM

|HEIGHT NUM CCOM WIDTH NUM CCOM

;

imagen: IMG SRC atrglobal ALT atrwh CBRACKET

|IMG atrglobal SRC ALT atrwh CBRACKET

|IMG atrglobal ALT SRC atrwh CBRACKET

|IMG atrglobal ALT atrwh SRC CBRACKET

|IMG SRC atrwh atrglobal ALT CBRACKET

|IMG atrwh SRC atrglobal ALT CBRACKET

|IMG atrwh atrglobal SRC ALT CBRACKET

|IMG atrwh atrglobal ALT SRC CBRACKET

|IMG SRC atrwh ALT atrglobal CBRACKET

|IMG atrwh SRC ALT atrglobal CBRACKET

|IMG atrwh ALT SRC atrglobal CBRACKET

|IMG atrwh ALT atrglobal SRC CBRACKET

|IMG SRC ALT atrwh CBRACKET

|IMG ALT SRC atrwh CBRACKET

|IMG ALT atrwh SRC CBRACKET

|IMG SRC atrwh ALT CBRACKET

|IMG atrwh SRC ALT CBRACKET

|IMG atrwh ALT SRC CBRACKET

|IMG SRC ALT CBRACKET

|IMG ALT SRC CBRACKET

|IMG SRC atrwh CBRACKET

|IMG atrwh SRC CBRACKET

|IMG SRC atrglobal CBRACKET

|IMG atrglobal SRC CBRACKET

|IMG SRC CBRACKET

;

**·*Código C***

yyerror(char \* msg)

{

printf("%s\n",msg);

}

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

{

printf("ANALISIS SINTACTICO\n");

if (argc==2)

{

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

if(yyin==NULL)

printf("\nNo se puede leer el archivo");

} else

{

printf("Bienvenido al MODO INTERACTIVO\n");

printf("Ingrese su codigo html\n");

printf("Si ha finalizado su codigo y no se detectaron errores,\n");

printf("ingrese ENTER y CTRL+ C para finalizar\n");

yyin = stdin;

yyparse ();

}

if( yyparse() == 0)

{

printf("\nCodigo CORRECTO\n");

printf("Para finalizar presione ctrl+c\n");

} else

{

printf("Error en la linea: %i\n",linea);

fclose(yyin);

return 0;

}

fclose(yyin);

getchar();

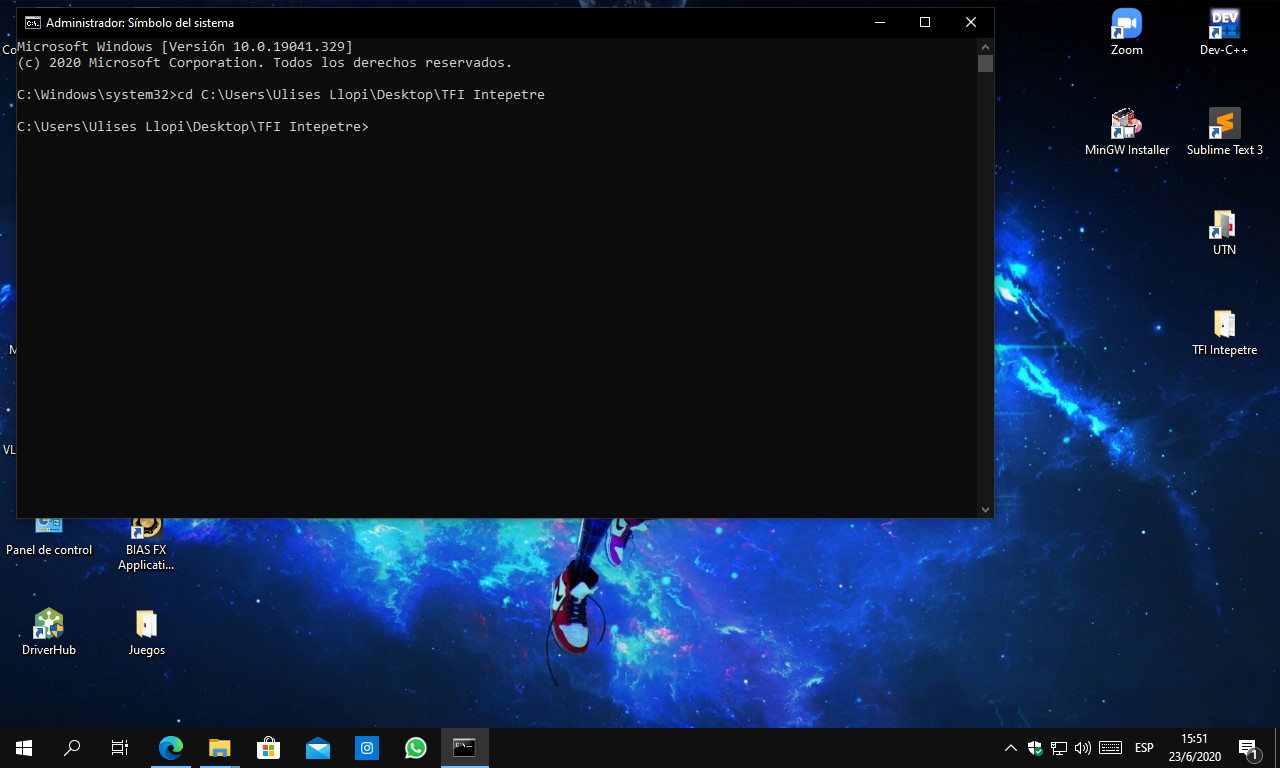
}

**Pasos para ejecutar el Parser:**

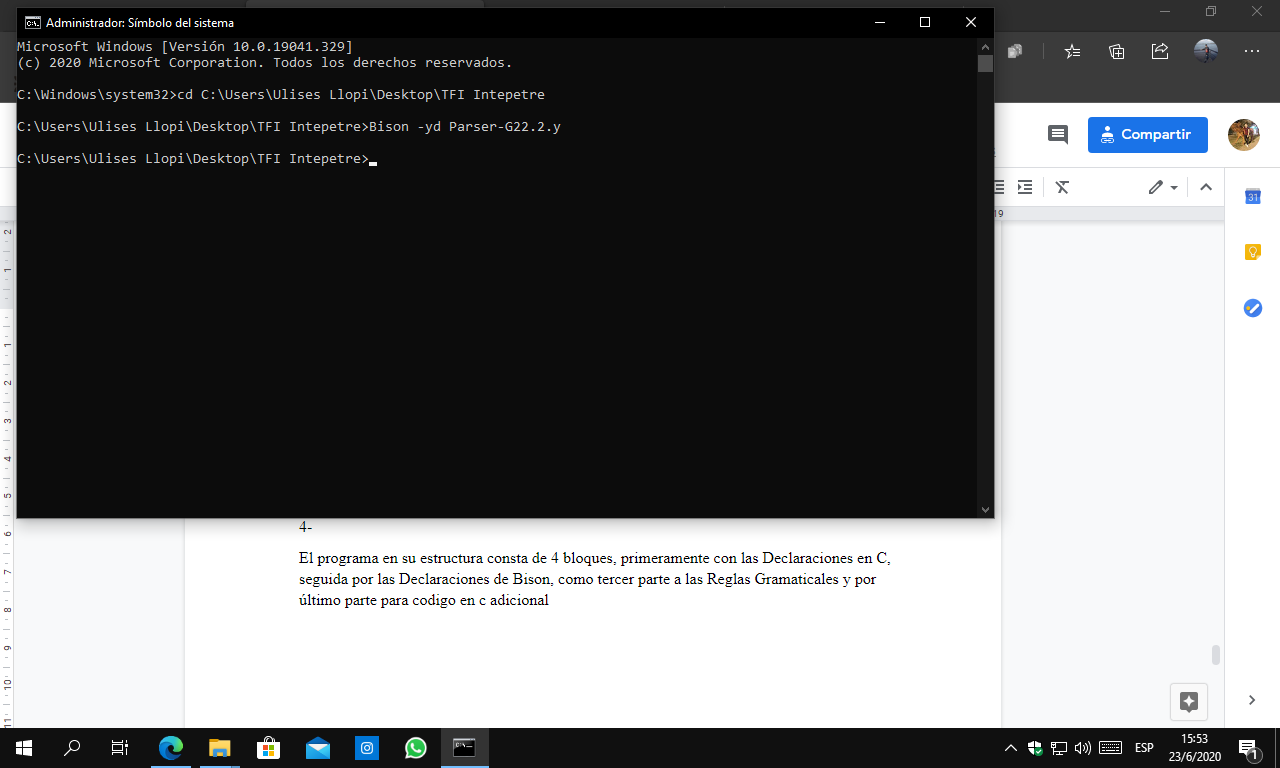
Se genera el código en C de la siguiente manera:

1- Se inicia la Consola de Windows o CMD ( también funciona con el Power Shell que utiliza Windows 10)

2- En la consola nos dirigiremos a la ubicación donde tenemos ubicado al archivo Parser.y



3- ejecutamos bison en la consola de la siguiente forma:

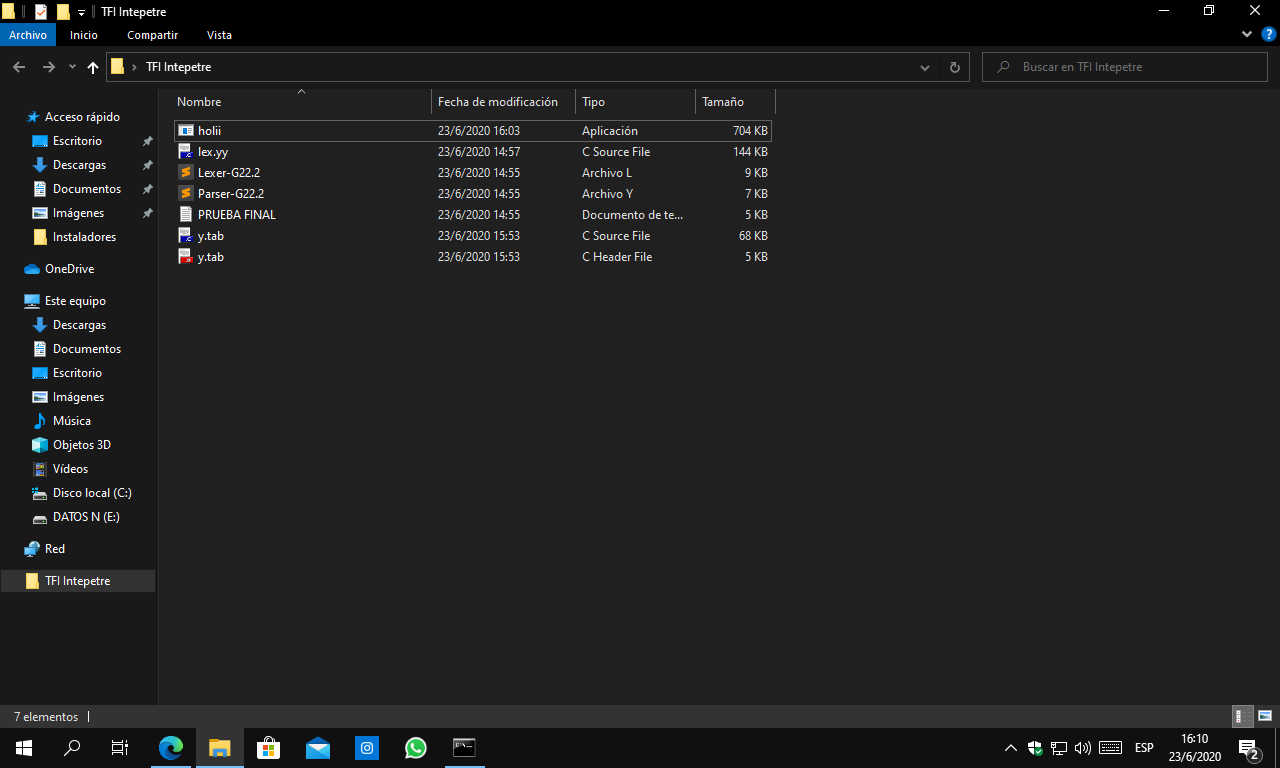


Con el nombre del Archivo Parser creado.

Se escribe -yd para que el archivo generado sea compatible con Yacc de Unix.

Una vez realizado estos pasos se nos generarán 2 archivos y.tab.c que nos servirán para el analizador sintáctico-léxico ejecutando la siguiente línea de código:

gcc y.tab.c lex.yy.c –lfl –o salida



Con esto ya tendremos el archivo ejecutable del Analizador.

# **FUNCIONES AUXILIARES**

Como función auxiliar del analizador el programa contiene un codigo en C en la seccion de Codigo adicional de C que es donde está definido el subprograma que tiene la finalidad de mostrar por pantalla textos informativos como por ejemplo mostrar la línea de código mal escrita así como también si se compila exitosamente, además de mostrar una combinación de teclas para salir del programa una vez finalizado la compilación exitosa del código como “Ctrl + D”.

La definición de yyerror en el parser nos sirve para mostrar un mensaje por pantalla los errores del código mal escrito. La variable yyparse que al estar mal escrito se pone en un valor distinto de 0, en cambio si es 0 el programa no presenta problemas.

Para contar las líneas de código se definió la variable “línea” que cada vez que el usuario avanza en una línea este se va incrementado.

# **MODO DE OBTENCIÓN DEL INTERPRETE**

Para obtener el analizador léxico se debe ejecutar en CMD “flex -i nombre\_del\_lexer.l” el cual nos generará un archivo lex.yy.c.

“Flex” como indica el comando compilara el archivo .l dado, el comando “-i” hará que nuestro lexer no diferencie letras mayusculas de minusculas.

Para el analizador sintáctico nuevamente en CMD se ingresa “bison -yd Nombre\_del\_Parser.y” que nos devolverá 2 archivos: el y.tab.c y el archivo y.tab.h

El comando Bison hará que compile el archivo .y cómo Parser, junto con el comando “-yd” haremos que este sea forzado a que el archivo generado sea compatible con Yacc de Linux,

* Nombre y descripción de cada archivo utilizado:

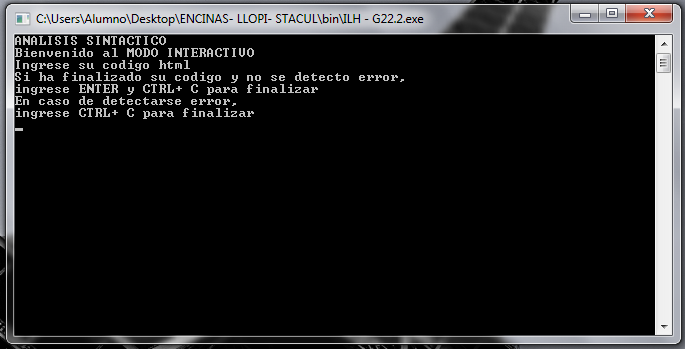
El archivo Lexer con extensión “.l” es el programa que contiene las el código en Flex que es la parte donde se define lo Léxico del programa, donde están definidas las expresiones regulares y la unión en este caso de las etiquetas del html con los Tokens que son utilizados en el Parser.

El archivo con extensión “.y” que es el parser está compuesto de tokens y la Gramática Libre de Contexto que define la estructura de el lenguaje elegido para interpretar el cual es compilado por CMD con el Bison que es el compilador del mismísimo Parser.

# **MODO DE EJECUCIÓN DEL INTERPRETE**

* **Manera Interactiva**

Para utilizar el programa como modo interactivo se debe ejecutar el archivo .exe el cual recibirá el código a escribir, éste mostrará el mensaje para que ingrese el código y para que una vez terminado el codigo ingrese la combinación de teclas anteriormente mencionada para salir del intérprete.



.

* **A partir de un archivo**

Para ejecutar el intérprete a partir de un archivo ya sea .html o .txt se debe ejecutar el archivo ejecutable via CMD (Simbolo del sistemas) de Windows seguida entre comillas el nombre del archivo con su extensión.

El archivo a ser analizado debe encontarse en la misma ubicacion que el .exe

# **EJEMPLOS**

***Ejemplo 1 Consigna.html***

<!DOCTYPE html>

<html lang="es">

<head>

<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">

<title>Presentacion de Interprete de Lenguajes HTML</title>

</head>

<body>

<SECTION id="intro">

<h2>UNIVERSIDAD TECNOLOGICA NACIONAL</h2>

<a href="http://www.frre.utn.edu.ar" target="\_blank">FACULTAD REGIONAL RESISTENCIA</a>

<h1>Sintaxis y Semántica de Lenguajes</h1>

<img src="http://frre.cvg.utn.edu.ar/pluginfile.php/29750/theme\_snap/coverimage/1584391474/course-image.gif" alt="imagen SSL" width="800" height="300">

<h2>Interprete de Lenguajes HTML</h2>

<strong>Trabajo Práctico Integrador:<br>Diseño e implementación de Lexer y Parser</strong>

<br>

<em>ciclo lectivo 2020</em>

<div id="indice">

<p>ir a seccion<a href="#elab">Elaboracion y entrega</a></p>

<p>ir a seccion<a href="#tags">Tags</a></p>

<p>ir a seccion<a href="#horarios">Horarios</a></p>

</div>

</section>

<h3>Objetivo</h3>

<p>elaborar un intérprete de lenguaje HTML (lenguaje de marcación de hipertexto) utilizando

algún lenguaje de programación o un generador de lexer y parser.</p>

<hr>

<section id="entrega">

<h3 id="elab">Elaboracion y Entrega</h3>

<ul>

<li><strong>Modo de realización del trabajo</strong>

<p>El trabajo se debe realizar de forma grupal en grupos de hasta

<strong>cuatro (4)</stRONG>alumnos, donde cada integrante deberá presentar una parte del

trabajo durante la presentación final.</p>

<p>Cada grupo estará representado por un delegado, encargado de subir

las entregas parciales y el trabajo final al campus virtual, enviará consultas o dudas,etc.</p>

</li>

<li><strong>Presentaciones</strong>

<p>Se preveen tres instacias de presentación:</p>

<ol type="I">

<li>1er entrega: Primera documentación del proyecto y gramática a

generar.<MarK>Viernes 24 de Abril de 2020</mark></li>

<li>2da entrega: Presentación del lexer o scanner que reconozca los

tokens del lenguaje.<mark>Viernes 27 de Mayo de 2020</mark></li>

<li>3er entrega final: Presentacion de trabajo completo. Lexer y

parser, incluye la presentación del trabajo ante la clase con una exposición de no más de 20

minutos.<mark><strong>Lunes 29 de Junio de 2020</strong></mark></li>

</ol>

</li>

</ul>

</section>

<hr>

<section id="tags">

<h3 id="elab">Principales Etiquetas</h3>

<ul>

<li><strong>Estructura documento</strong>

<p>Describen la estructura general del documento:</p>

<ol type="a">

<li>DOCTYPE</li>

<li>html</li>

<li>head</li>

<li>title</li>

<li>body</li>

<li>section</li>

<li>div</li>

</ol>

</li>

<li><strong>Etiquetas de parrafo</strong>

<p>Describen texto:</p>

<ol type="a">

<li>p</li>

<li>h1 a h2</li>

<li>br</li>

<li>hr</li>

<li>strong</li>

<li>em</li>

<li>mark</li>

</ol>

</li>

<li><strong>Listas</strong>

<p>Las listas permiten crear párrafos agrupados y alineados, como este

:)</p>

<ol type="a">

<li>ul</li>

<li>ol</li>

<li>li</li>

</ol>

</li>

<li><strong>Tablas</strong>

<p>Una tabla es un conjunto de datos presentados en forma de filas y

columnas.</p>

<ol type="a">

<li>table</li>

<li>tr</li>

<li>td</li>

<li>th</li>

<li>thead</li>

<li>tfoot</li>

<li>tbody</li>

</ol>

</li>

<li><strong>Enlaces</strong>

<p>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.</p>

<P>La sintaxis completa de una URL es la siguiente:<MARK>

protocolo://dominio:puerto/ruta#fragmento</mark></p>

</li>

<li><strong>Imagenes</strong>

<p>Las imágenes se colocan mediante la etiqueta img. El atributo src indica

la URL (relativa o absoluta) a la imagen</p>

</li>

</ul>

</section>

<hr>

<section id="horarios">

<table border="1">

<thead>

<tr>

<td>2.1</td>

<th>Practica</th>

<th>Teoria</th>

</tr>

</thead>

<tbody>

<tr>

<th>Jueves</th>

<td>18:30</td>

<td></td>

</tr>

<tr>

<th>Viernes</th>

<td>21:30</td>

<td></td>

</tr>

<tr>

<th>Miercoles</th>

<td></td>

<td>14:15</td>

</tr>

</tbody>

</table>

</section>

</body>

</html>

***saba.html***

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" >

<meta name="Analizador" content="Sintactico">

<TITLE>SOY SABALERO</TITLE>

</head>

<BODY>

<a href="https://www.youtube.com/watch?v=PTjuYrZPq7U"><h1> SOY SABALERO </h1> </a>

<p> No hace falta que les digan que soy raza <br>

Por mi piel escapa el alma sabalera,</p>

<h3>Sangre y luto es el color de mi bandera</h3>

<SECTION>Y a los negros yo los llevo aquí en el alma.</section>

<br>

<ol type="a">

<li>Es el negro un sentimiento que se lleva de verdad</li>

<li>Y que late al compás del corazón,</li>

<li>Si ganara yo al infierno por ser negro y nada más</li>

<li>Moriré llevando negro el corazón.</li>

</ol>

<ul>

<li>Sabale, sabale, la culpa de este pueblo sabalero,</li>

<li>Sabale, sabale, que todo el mundo grite dale negro!</li>

</ul>

<hr>

<strong class="AH EH EH AH"> yo soy SABALERO,</strong>

<strong class="AH EH EH AH" id="sabalero">, SABALERO </strong>

<em> AH EH EH AH yo soy SABALERO,</EM>

<MARK>AH EH EH AH sabalero, Sabalero.</MARK>

<hr>

<table>

<tr>

<td> <strong>He nacido en las orillas roja y negras</strong></td>

<td><strong>Yo me entiendo con la gente sabalera,.</strong></td>

<th><strong>Soy amiga del ciruja y del maestro</strong> </th>

<td> <strong> Con el tordo y con el punga yo me encuentro. </strong> </td>

</tr>

</table>

<br>

<ol>

<li>Y en mi casa se codea la pobreza y el señor,</li>

<li>Con el cura el judío y el pastor,</li>

<li>Aunque me ganara el cielo por cambiarme de color</li>

<li>Moriré llevando <strong>negro el corazón.</strong></li>

</ol>

<br>

Sabale, sabale, la culpa de este pueblo sabalero,

Sabale, sabale, que todo el mundo grite dale negro!

<br>

<em> Ah eh eh ah yo soy sabalero, </em>

<em> Ah eh eh ah sabalero, Sabalero, </em>

<em> Ah eh eh ah yo soy sabalero, </em>

<em> Ah eh eh ah sabalero, Sabalero. </em>

<em> Ah eh eh ah yo soy sabalero</em>

<em> Ah eh eh ah sabalero, Sabalero</em>

</body>

</html>

***Error Atributo.html***

<!DOCTYPE html>

<html lang="pr">

<head>

<meta charset="UTF-8" >

<meta name="ejemplo" content="un ejemplo de html">

<TITLE>Página de ejemplos de HTML</TITLE>

</HEAD>

<BODY>

<H1 id="uno" id="dos">Página de ejemplos de HTML</H1>

<H1 height="45">Cabecera tipo 1</H1>

<H3 border="9">Cabecera tipo 3</H3>

<H5>Cabecera tipo 5</H5>

<H6>Cabecera tipo 6</H6>

<IMG SRC="bescut.gif" ALT="Escudo UPV" height="803">Ejemplo de imagen</IMG>

</BODY>

</HTML>

***Ejemplo TFI.html***

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<title> HOLA! </title>

</head>

<body>

<p> El trabajo final integrador consta de un interprete de lenguaje en este caso de HTML5

Para la primera instancia elaboramos la gramatica que genera dicho lenguaje, que mas tarde seguiria nuestro Parser para realizar el analisis sintactico

</p>

<p> Como segunda instancia propuesta por la catedra, elaboramos el analizador lexico compuesto primeramente por las expresiones regulares a utilizar seguidas de las reglas </p>

<h2> Para poder ejecutar y compilar el analizador lexico </h2>

<p> se requiere tener descargado Flex, la version que utilizamos fue: 2. 5.35 </p>

<p>En la tercera instancia y con el Lexer aprobado, pasamos a la realizacion del analizador sintactico Parser que consta de las declaraciones de los tokens que recibe del Lexer y en segundo lugar las Reglas Gramaticales, que son las que definen la estructura que debe seguir el lenguaje

</p>

<p> Lista de programas utilizados </p>

<ol>

<li> Flex 2.5.35</li>

<li> Al compilar el archivo "Lexer-G22.2.l" con "Flex" desde los comandos de sistema se genera el archivo "lex.yy.c" conteniendo el codigo necesario en <STRONG>C</STRONG> </li>

<li> Bison 2.4.2 </li>

<li> una vez compilado el archivo " Parser-G22.2.y" con Bison junto a la opcion -yd se generan dos archivos: y.tab.c e y.tab.h. </li>

<li> Al contar con el archivo lex.yy.c junto con archivo y.tab.c, con la instruccion gcc obtenemos el Interprete de Lenguaje HTML5 </li>

</ol>

<p> <strong> Este interprete cuenta con dos restricciones: </strong> </p> <br>

<ul>

<li> No reconoce espacios en las etiquetas de apertura, es decir, del lado derecho del tag, pero si del lado izquierdo </li>

<li> No reconoce letras con tildes al analizar un archivo, pero si al ingresar desde el <strong>modo interactivo</strong></li>

<li> <mark>Sin embargo esto no afecta al correcto funcionamiento del Analizador Sintactico </mark></li>

</ul>

<p> Esperamos que les guste nuestro trabajo tanto como a nosotros! </p>

</body>

</html>

# **CONCLUSIÓN**

Con este trabajo hemos comprendido la importancia de lo dado en clases de “Sintaxis y Semántica de los Lenguajes”, viendo la aplicación de las expresiones regulares como los árboles de derivación, que nos permitieron eliminar las ambigüedades presentes en las primeras instancias de nuestra gramática.

Tuvimos que familiarizarnos con varios programas que hasta ese entonces eran ajenos a nosotros, Flex y Bison como también el manejo de los comandos de sistema.

Se nos presentaron varias complicaciones, pero rescatamos el hecho de haber podido solucionarlas a todas. A pesar de ser un desafío en un ámbito tan desconocido para nosotros, estamos muy satisfechos con el resultado obtenido y con los conocimientos que hemos adquirido en todo el proceso.

Respecto a los puntos fuertes de nuestro Lexer destacamos la veracidad para identificar las palabras reservadas del lenguaje como también de nuestro Parser al ser capaz de distinguir el orden correcto de las producciones ingresadas. Ademas, al ser un archivo .exe se necesita muy poca memoria y tiempo para su ejecución.

Al hablar de sus puntos débiles nos encontramos con Flex y Bison que son programas un poco anticuados a la hora de tratar la interfaz gráfica ya que la manera en que presenta la información es a través de los comandos de sistema viéndose limitado a solo colores blanco y negro.

# **BIBLIOGRAFÍA**

* Archivo PDF con especificaciones para la realización del trabajo [https://frre.cvg.utn.edu.ar/pluginfile.php/82337/mod\_resource/content/3/SSL2020TP1\_InterpreteLenguajeHTML.pdf](https://frre.cvg.utn.edu.ar/pluginfile.php/82337/mod_resource/content/3/SSL2020-TP1_InterpreteLenguajeHTML.pdf)

Manuales de Flex y Bison

* http://webdiis.unizar.es/asignaturas/LGA/