UNIVERSIDAD TECNOLOGICA NACIONAL

Sintaxis y Semántica del Lenguaje

Trabajo Practico Integrador

Diseño e implementación

De Lexer y Parser.

ING. EN SISTEMAS DE INFORMACION

Grupo 213

Fernandez, Ian.

Kalbermatter, Alan.

Alegre, Jose.

Orban, Joel.

2020

INDICE

Introducción

Competencias

Para el presente trabajo practico se pretende desarrollar competencias tales en la asignatura como:

*Comprender los fundamentos teóricos de los lenguajes de programación y las distintas técnicas del diseño y procesamiento lexico, gramatical y sintáctico. Importante remarcar que fue adquirido conocimiento para aplicarlas en la creación y desarrollo de lenguajes de programación.

Los primeros compiladores en la década de los 50's usaban técnicas "ad hoc" para analizar el código fuente de programas que eran compilados. Durante los 60's, el campo obtuvo gran atención academica, y por sobre los primeros años de la década del 70, el análisis sintactico ya era un campo bien comprendido.

Una de las ideas clave fue la de partir el trabajo en dos partes: análisis lexico(también llamado lexing o escaneo) y el análisis sintactico (o parsing).

El punto de esta introducción se debe para introducir la idea histórica y los conceptos teóricos a los objetivos del trabajo desarrollado.

*Fragmento extraído de la bibliografía Flex & Bison – Unix by John R. Levine.

Objetivo

Diseñar un interprete del lenguaje HTML 5 mediante la implementacion de un lenguaje de programación en conjunto con un generador lexico y parser.

Descripcion de como se implemento la solución

Para alcanzar el objetivo principal propuesto, anteriormente se llevaron a cabo dos etapas no menos importantes en el área y una tercera y final donde se culmino con la integración total de las partes.

Durante la primera etapa se diseño una gramática clasificada como libre de contexto, la cual debería describir el lenguaje solicitado(HTML con ciertas restricciones y especificaciones).

Continuando con la segunda etapa en la cual se inicio con una investigación de las herramientas dispuestas por la catedra para el análisis lexico del lenguaje, se identificaron diferentes lenguajes en los cuales estas herramientas se desenvolvían. Tras un análisis de recursos tomamos la decisión de implementar el uso de expresiones regulares y la técnica LALR a través de la implementacion libre y gratuita de Lex y Yacc, Flex y Bison, al deberse el hecho de que estos códigos generados eran en el lenguaje C y que el ancho del grupo tenia previo conocimiento en el mismo.

Como tercera y ultima entrega, se implementaron las anteriores mencionadas y se diseño una gramática dentro de la herramienta Yacc, donde a través de técnicas especificas y métodos logramos la implementacion tanto de Lexer como de Yacc mediante la compilación en el lenguaje C de ambas partes.

Para entender mejor dichas herramientas, realizamos un estudio de las mismas para poder comprenderlas y luego aplicarlas dentro de nuestro trabajo.

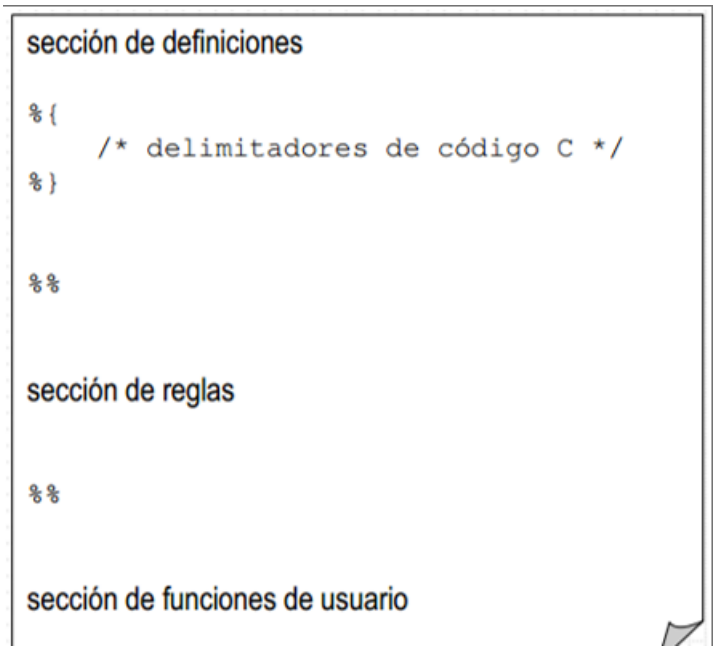
A continuación describiremos las mismas, además de una introducción de la implementacion de estas.

Herramienta Flex utilizada para la creación del Lexer o Escaner Lexico.

Flex es una herramienta la cual permite el análisis lexico a partir de un texto o formato cadena descripto. Esta herramienta busca coincidencias a partir de un fichero de entrada y ejecuta acciones asociadas a estas expresiones.

Para poder obtener el scanner es requerido un fichero del formato extensión .l que se llama al ejecutar desde la terminal el comando *flex -i [nombre del .l]*, este comando tiene como salida un archivo extensión .C.

Durante el codificado de la herramienta para su implementacion se separa en tres secciones, separadas por una línea donde aparece un '%%' en la siguiente distribución:



En la sección de definiciones se especifican las definiciones de nombres y las declaraciones de condiciones de arranque.

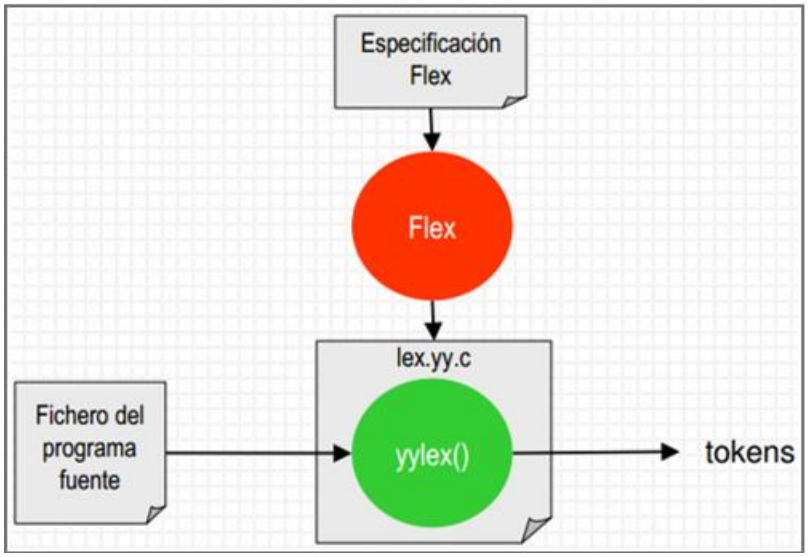
La sección de relgas posee una serie de reglas de la forma *expresión regular – token.*

Para finalizar, la sección de código posee un main que sencillamente se copia al archivo fuente 'lex.yy.c'. Este mismo realiza rutinas de escaneo las cuales en complementación llaman al scanner o son llamadas por este.

Durante la compilación del fichero .l, se genera el fichero fuente .C ya mencionado anteriormente, el cual define una función 'yylex()', el cual se compila y es enlazado con la librería de Flex para producir un archivo ejecutable del tipo .exe.

El programa lexer va recorrriendo el texto y transformando el texto, Flex lee los ficheros de entrada que le son proporcionados por el usuario de varias formas las cuales son a elección de este.

El sistema de trabajo de esta herramienta avanza en el texto y matchea o empareja reglas que coincidan con el texto, si no encuentra ninguna coincidencia devuelve el mismo texto que recibió como entrada. Si encuentra dos o mas emparejamientos, toma el que empareje el texto mas largo. A su vez, si el mismo encuentrar mas de dos emparejamientos de la misma longitud, se escoge la regla listada en primer lugar del fichero de entrada de Flex. Una vez finalizado el emparejamiento, el texto correspondiente al emparejamiento (token) se encontrara disponible en un puntero de carácter global de nombre yytext como su longitud en la variable global entera yyleng.



Herramienta bison, descripción para la implementacion del analizado sintactico

Bison toma la gramática que le fue especificada y escribe un parser que determina oraciones validas en es dicha gramática. Al ser un generado de analizadores sintácticos de propósito general que convierte una descripción para una gramática independiente del contexto (en realidad de una subclase de las mismas, LALR) en programa en lenguaje C el cual analiza dicha gramática.

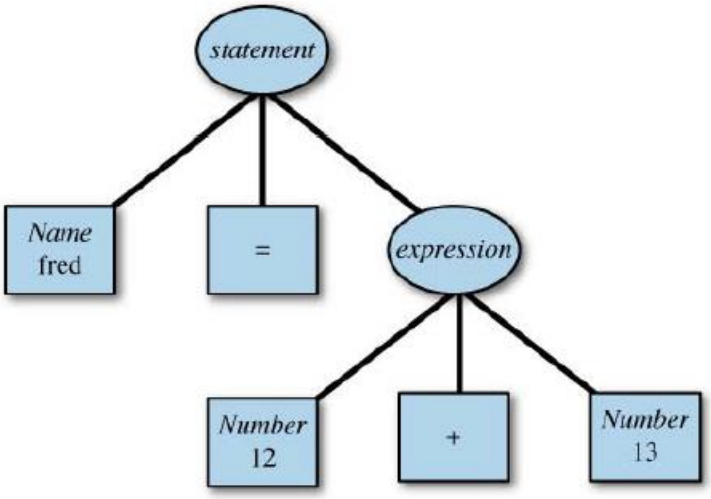
El fichero funete generado por Bison (de extensión del tipo .y) se desenvuelve describiendo una gramática. La forma general de fichero del tipo Bison es la siguiente:

```
%{
declaraciones en C
%}
Declaraciones de Bison
%%
Reglas gramaticales
%%
Código C adicional
```

Las declaraciones de Bison declaran los nombres de los símbolos terminales y no terminales, y también podrían describir la precedencia de operadores y los tipos de datos de los valores semánticos de varios símbolos.

Las reglas gramaticales son las producciones de la gramática, que además pueden llevar asociadas acciones, código en C, que se ejecutan cuando el analizador encuentra las reglas correspondientes.

Para entender mejor la forma en la que se desarrolla esta herramienta a la hora de implementarla, la misma describe un árbol de derivaciones del tipo de la imagen siguiente



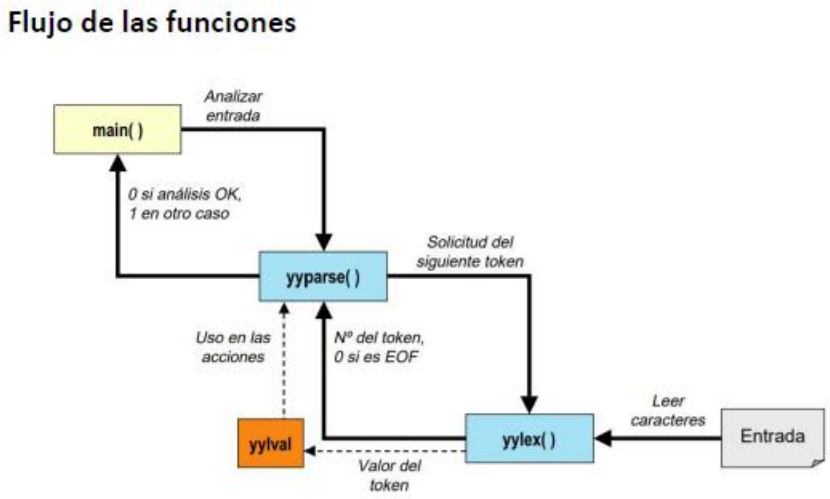
el cual muestra como a través de nodos terminales(tokens) y no terminales (reglas) se puede desarrollar cadenas de salida las cuales se van emparejando en el texto de entrada para poder determinar su correcta expresión en caso de coincidencia.

Interacciones Flex-Bison

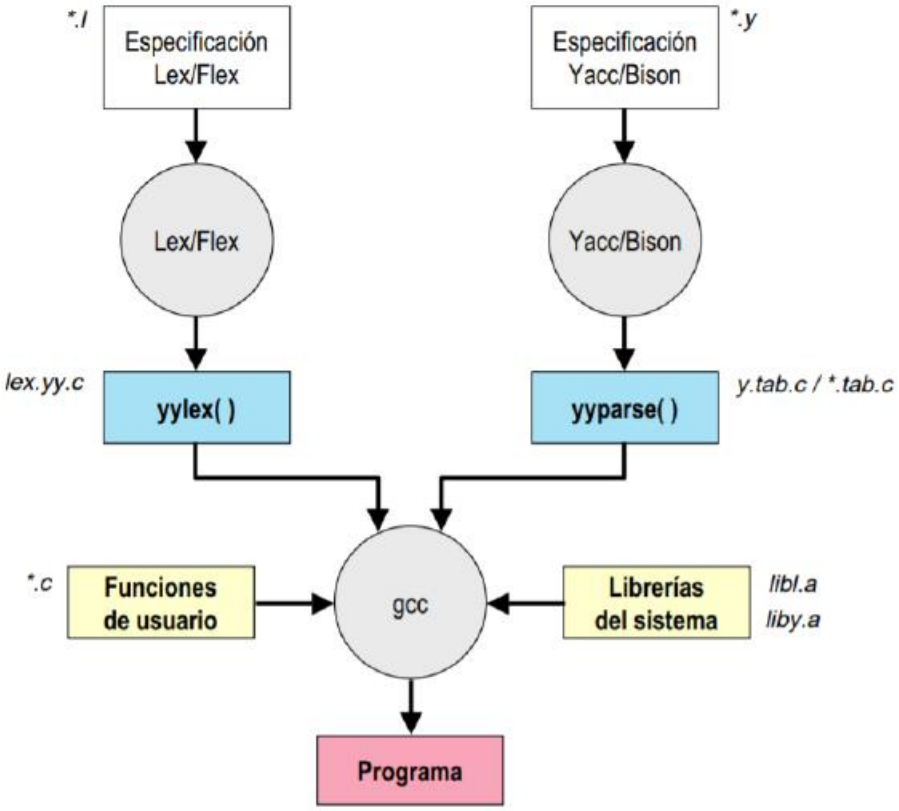
Bison utiliza una función llamada ‘yylex()’ para devolver el siguiente token de la entrada, además pueden poner cualquier valor asociado en la variable global ‘yyval()’. Para poder implementar estas herramientas, a la hora de compilar el fichero de extensión .y con Bison, este ultimo genera dos nuevos ficheros, uno es un archivo del tipo ‘y.tab.c’ el cual es utilizado a la hora de compilar el conjunto y otro del tipo ‘y.tab.h’, este ultimo contiene las definiciones de todos los tokens que se encuentran en el archivo fuente de Bison. Este fichero de cabecera es incluido luego en el fichero fuente de Flex.

Funcionamiento del analizador sintactico

El archivo fuente de Bison es convertido en una función llamada yyparse. Esta es llamada para hacer que el análisis comience. Esta función lee tokens, ejecuta acciones de sintaxis, y por ultimo retorna cuando se encuentra al final del fichero analizado o cuando es encontrado un error de sintaxis del cual no puede recuperarse. El valor devuelto por la función en caso de que el análisis sea exitoso es 0, en caso de fallas durante el análisis, es 1.



A continuación mostramos un esquema de creación el cual detalla específicamente la labor de las herramientas explicadas.



Las funciones, librerías utilizadas en la herramienta son:

- yywrap(): es llamado por lex cuando la entrada está agotada (o en final de archivo).
- #define YYDEBUG 1: El mismo permite inicializar el contador igual a 1 al compilar el analizador sintáctico.
- #include "y.tab.h" : Utilizado en el archivo del analizador léxico, le estamos incluyendo el archivo y.tab.h que son las funciones del analizador sintáctico.
- %option caseless : Permite al analizador Léxico ignorar la diferencia entre los caracteres de tipo mayúscula y minúscula, vuelve al lenguaje analizador “No sensitive Case”.

- %option yylineno : Es una opción utilizada en el analizador léxico, permite contar las líneas analizadas en el programa.
- #include<time.h> y #include <string.h> : Son utilizadas para las funciones descriptas en el apartado de funciones del usuario en el archivo del analizador sintáctico. Time permite generar un valor aleatorio el cual lo utilizamos para ingresar de forma indexada al arreglo y asi obtener una respuesta que es emitida por el parser al finalizar su análisis.
- #include <stdio.h> , #include <stdlib.h> Librerias que nos permiten funcionalidades básicas del programa, tales como por ejemplo imprimir por pantalla.

Gramática

sigma : **DOCTYPE HTML** head cuerpo **C_HTML;**

head : **A_HEAD** cont_cabeza;

cont_cabeza : **METAS** enc;

enc : **META_CHARSET** enc | **C_HEAD** | **A_TITLE** atrib **ENTRADA** C_TITLE enc;

cuerpo : **A_BODY** atrib otrocuerpo;

otrocuerpo : seccion | division | funciones | **ENTRADA** | **C_BODY**;

seccion : **A_SECCION** atrib funciones **C_SECCION** otrocuerpo

| **A_SECCION** atrib funciones **C_SECCION**;

division : **A_DIV** atrib funciones **C_DIV** otrocuerpo

| **A_DIV** atrib funciones **C_DIV**;

funciones : encabezado | parrafo | em | strong | mark | br | hr | lista | tabla | a
| img | selector;

selector : seccion | division;

img : **A_IMG** cont_img otrocuerpo | **A_IMG** cont_img **ENTRADA** otrocuerpo;

cont_img : atribglb **SRC** atrib_img **LC**

| atribglb atrib_img **SRC LC**

| **SRC** atrib_img atribglb **LC**

| **SRC** atribglb atrib_img **LC**

| atrib_img **SRC** atribglb **LC**

| atrib_img **SRC** atrib_img **LC**

| atrib_img atribglb **SRC LC**

| **SRC** atrib_img **LC**

| atrib_img **SRC LC**

| **SRC** atribglb **LC**

| atribglb **SRC LC**

| **SRC LC**;

atrib_img : **ALT WIDTH HEIGHT**

| **ALT HEIGHT WIDTH**

| **WIDTH ALT HEIGHT**

| **WIDTH HEIGHT ALT**

| **HEIGHT ALT WIDTH**

| **HEIGHT WIDTH ALT**

| **ALT WIDTH**

| **WIDTH ALT**

| **ALT HEIGHT**

| **HEIGHT ALT**

| **HEIGHT WIDTH**

| **WIDTH HEIGHT**

| **ALT**

| **HEIGHT**

| **WIDTH**;

a : **A_A** cont_enlace **C_A** otrocuerpo

| **A_A** cont_enlace **ENTRADA C_A** otrocuerpo

| **A_A** cont_enlace **ENTRADA C_A** | **A_A** cont_enlace **C_A**

| **A_A** cont_enlace funciones **C_A**

| **A_A** cont_enlace funciones **C_A** otrocuerpo;

cont_enlace : atribglb **ENLACE** atrib_enlace **LC**

| **ENLACE** atribglb atrib_enlace **LC**

| atrib_enlace **ENLACE** atribglb **LC**

| **ENLACE** atrib_enlace atribglb **LC**

| **ENLACE** atrib_enlace **LC**

| atrib_enlace **ENLACE LC**

| **ENLACE LC**

| atrib_enlace **LC**

| atribglb **ENLACE LC**

| **ENLACE** atribglb **LC**

| **LC**;

tabla : **A_TABLA** opciones_t elementos_t **C_TABLA**

| **A_TABLA** opciones_t elementos_t **C_TABLA** otrocuerpo;

opciones_t : **LC** | **BORDER RULES LC** | **RULES BORDER LC** | **RULES LC** | **BORDER LC**;

elementos_t : **A_CAPTION** atrib **ENTRADA C_CAPTION** parte_tab | parte_tab;

parte_tab : tabla_i

| t_cabeza t_cuerpo t_pies

| t_cuerpo t_cabeza t_pies

| t_cuerpo t_pies t_cabeza

| t_cabeza t_pies t_cuerpo

| t_pies t_cabeza t_cuerpo

| t_cabeza t_cuerpo

| t_cabeza t_pies

| t_cuerpo t_cabeza

| t_pies t_cabeza;

t_cabeza : **A_TBH** tabla_i **C_TBH**

t_cuerpo : **A_TBB** tabla_i **C_TBB**

t_pies : **A_TBF** tabla_i **C_TBF**;

tabla_i : **A_TR** tablas;

tablas : **A_TH ENTRADA C_TH** tablas

| **A_TD** ENTRADA **C_TD** tablas

| **A_TH C_TH** tablas

| **A_TD C_TD** tablas

| **C_TR** tabla_i

| **C_TR**;

lista : **A_UL** atrib **ENTRADA** funciones **C_UL** otrocuerpo

| **A_UL** atrib **ENTRADA** funciones **C_UL**

| **A_UL** atrib funciones **ENTRADA C_UL** otrocuerpo

| **A_UL** atrib funciones **ENTRADA C_UL**

| **A_UL** atrib **ENTRADA C_UL**

| **A_UL** atrib **ENTRADA C_UL** otrocuerpo

| **A_UL** atrib funciones **C_UL**

| **A_UL** atrib funciones **C_UL** otrocuerpo

| **A_LI** atribli **ENTRADA** funciones **C_LI** otrocuerpo

| **A_LI** atribli **ENTRADA** funciones **C_LI**

| **A_LI** atribli funciones **ENTRADA C_LI** otrocuerpo

| **A_LI** atribli funciones **ENTRADA C_LI**

| **A_LI** atribli **ENTRADA C_LI**

| **A_LI** atribli **ENTRADA C_LI** otrocuerpo

| **A_LI** atribli funciones **C_LI**

| **A_LI** atribli funciones **C_LI** otrocuerpo

| **A_OL** atriblo **ENTRADA** funciones **C_OL** otrocuerpo

| **A_OL** atriblo **ENTRADA** funciones **C_OL**

| **A_OL** atriblo funciones **ENTRADA C_OL** otrocuerpo

| **A_OL** atriblo funciones **ENTRADA C_OL**

| **A_OL** atriblo **ENTRADA C_OL**

| **A_OL** atriblo **ENTRADA C_OL** otrocuerpo

| **A_OL** atriblo funciones **C_OL**

| **A_OL** atriblo funciones **C_OL** otrocuerpo;

atribli :**LC** | atribglb **LC** | **VALUE LC** | atribglb **VALUE LC**;

atriblo :**LC** | atribglb **LC** | **TYPE LC** | atribglb **TYPE LC**;

atribglb :**ID** | **CLASS** | **ID CLASS** | **CLASS ID**;

atrib :**ID LC** | **CLASS LC** | **ID CLASS LC** | **CLASS ID LC** | **LC**;

parrafo : **A_P** atrib **ENTRADA** funciones **C_P** otrocuerpo

| **A_P** atrib **ENTRADA** funciones **C_P**

| **A_P** atrib funciones **ENTRADA C_P** otrocuerpo

| **A_P** atrib funciones **ENTRADA C_P**

| **A_P** atrib **ENTRADA C_P**

| **A_P** atrib **ENTRADA C_P** otrocuerpo

| **A_P** atrib funciones **C_P**

| **A_P** atrib funciones **C_P** otrocuerpo;

em : **A_EM** atrib **ENTRADA** funciones **C_EM** otrocuerpo

| **A_EM** atrib **ENTRADA** funciones **C_EM**

| **A_EM** atrib funciones **ENTRADA C_EM** otrocuerpo

| **A_EM** atrib funciones **ENTRADA C_EM**

| **A_EM** atrib **ENTRADA C_EM**

| **A_EM** atrib **ENTRADA C_EM** otrocuerpo

| **A_EM** atrib funciones **C_EM**

| **A_EM** atrib funciones **C_EM** otrocuerpo;

strong : **A_STRONG** atrib **ENTRADA** funciones **C_STRONG** otrocuerpo

| **A_STRONG** atrib **ENTRADA** funciones **C_STRONG**

| **A_STRONG** atrib funciones **ENTRADA C_STRONG** otrocuerpo

| **A_STRONG** atrib funciones **ENTRADA C_STRONG**

| **A_STRONG** atrib **ENTRADA C_STRONG**

| **A_STRONG** atrib **ENTRADA C_STRONG** otrocuerpo

| **A_STRONG** atrib funciones **C_STRONG**

| **A_STRONG** atrib funciones **C_STRONG** otrocuerpo;

mark : **A_MARK** atrib **ENTRADA** funciones **C_MARK** otrocuerpo

| **A_MARK** atrib **ENTRADA** funciones **C_MARK**

| **A_MARK** atrib funciones **ENTRADA C_MARK** otrocuerpo

| **A_MARK** atrib funciones **ENTRADA C_MARK**

| **A_MARK** atrib **ENTRADA C_MARK**

| **A_MARK** atrib **ENTRADA C_MARK** otrocuerpo

| **A_MARK** atrib funciones **C_MARK**

| **A_MARK** atrib funciones **C_MARK** otrocuerpo;

br : **A_BR** atrib otrocuerpo | **A_BR** atrib otrocuerpo **C_BODY** ;

hr : **A_HR** atrib otrocuerpo | **A_HR** atrib otrocuerpo **C_BODY** ;

encabezado : **H1** atrib **ENTRADA C_H1 ENTRADA** otrocuerpo

| **H1** atrib **ENTRADA C_H1**

| **H1** atrib **ENTRADA C_H1** otrocuerpo

| **H2** atrib **ENTRADA C_H2 ENTRADA** otrocuerpo

| **H2** atrib **ENTRADA C_H2**

| **H2** atrib **ENTRADA C_H2** otrocuerpo

| **H3** atrib **ENTRADA C_H3 ENTRADA** otrocuerpo

| **H3** atrib **ENTRADA C_H3**

| **H3** atrib **ENTRADA C_H3** otrocuerpo

| **H4** atrib **ENTRADA C_H4 ENTRADA** otrocuerpo

| **H4** atrib **ENTRADA C_H4**

| **H4** atrib **ENTRADA C_H4** otrocuerpo

| **H5** atrib **ENTRADA C_H5 ENTRADA** otrocuerpo

| **H5** atrib **ENTRADA C_H5**

| **H5** atrib **ENTRADA C_H5** otrocuerpo

| **H6** atrib **ENTRADA C_H6 ENTRADA** otrocuerpo

| **H6** atrib **ENTRADA C_H6**

| **H6** atrib **ENTRADA C_H6** otrocuerpo;

atrib_enlace : **BLANK**

| **PARENT**

| **SELF**

| **TOP**

| **BLANK PARENT**

| **PARENT BLANK**

| **BLANK SELF**

| **SELF BLANK**

| **SELF PARENT**

| **PARENT SELF**

| **BLANK PARENT SELF**

| **BLANK SELF PARENT**

| **PARENT BLANK SELF**

| **PARENT SELF BLANK**

| **SELF BLANK PARENT**

| **SELF PARENT BLANK**;

Símbolos no Terminales

DOCTYPE HTML → <!DOCTYPE html>
A_HEAD →<head>
META_CHARSET →<meta name="{ContMeta}"content="\"{Ent}\"">
METAS →<meta charset="{UTF}">
C_HEAD →</head>
A_TITLE →<title
ENTRADA → Entrada de texto por teclado
C_TITLE →</title>
ID →id=\"[a-zA-Z0-9-]
CLASS →class=\"[a-zA-Z0-9-]*\
LC →>
C_HTML→</html>
A_BODY →<body
C_BODY →</body>
A_SECCION →<section
A_DIV →<div
C_SECCION→</section>
C_DIV →
H1 →<h1
H2 →<h2
H3 →<h3
H4 →<h4
H5 →<h5
H6 →<h6
C_H1 →</h1>
C_H2→</h2>
C_H3 →</h3>
C_H4 →</h4>
C_H5 →</h5>

C_H6→</h6>
A_P →<p
C_P →</p>
A_EM →<em
C_EM →
A_STRONG →<strong
C_STRONG →
A_MARK →<mark
C_MARK →</mark>
A_BR →<br
C_BR →>
A_HR →<hr
C_HR →>
A_UL →<ul
C_UL →
A_LI →<li
C_LI →
A_OL →<ol
C_OL →
VALUE→value=\"[0-9]*\"
TYPE→type=\"[a-zA-Z0-9-]*\"
A_TABLA →<table
C_TABLA →</table>
BORDER →border=\"[0-9]\"
RULES →rules=\"[a-zA-Z0-9]*\"
A_CAPTION →<caption
C_CAPTION →</caption>
A_TBH →<thead>
A_TBB →<tbody>
A_TBF →<tfoot>
C_TBH →</thead>

C_TBB→</tbody>
C_TBF →</tfoot>
A_TR →<tr>
C_TR →</tr>
A_TH →<th>
C_TH →</th>
A_TD →<td>
C_TD→</td>
ENLACE →{href}{Enlace}|{href}\""#"{Ent}\"
BLANK →target=\"_blank\"
PARENT →target=\"_parent\"
SELF →target=\"_self\"
C_A →
A_A TOP→target=\"_top\"

Algunas expresiones no se pueden completar debida a su alta complejidad en símbolos.

Expresiones regulares

comilla [\'|\"]+

Ent [_a-zA-Z0-9À-ÿ|\!|\?|\$|\%|\&|\||\.|\:|\/|_|*|\(|\=|\)|\{|\}|\[|\]|\+|\@|\^|\#|\;|\,|u00f1]*

REG_ID ([ ]*id=\"[a-zA-Z0-9-]*\"[ ]*)

REG_CLASS ([ ]*class=\"[a-zA-Z0-9-]*\"[ ]*)

Alistaol ([ ]*type=\"[a-zA-Z0-9-]*\"[ ]*)

REG_BORDER ([ ]*border=\"[0-9]\"[ ]*)

REG_RULES ([ ]*rules=\"[a-zA-Z0-9]*\"[ ]*)

Alistali ([ ]*value=\"[0-9]*\"[ ]*)

REG_BLANK ([ ]*target=\"_blank\"[ ]*)

REG_PARENT	([ ]*target=\"_parent\"[ ]*)
REG_SELF	([ ]*target=\"_self\"[ ]*)
REG_TOP	([ ]*target=\"_top\"[ ]*)
REG_ALT	([ ]*alt=\"[a-zA-Z0-9]*\"[ ]*)
REG_WIDTH	([ ]*width=\"[0-9]*\"[ ]*)
REG_HEIGHT	([ ]*height=\"[0-9]*\"[ ]*)
Enlace	\"({Prot}{URL}{Term}*)\"
Term	.[a-z]{2,3}|:[0-9]*|[/[_a-zA-Z0-9]*]*|#[_a-zA-Z0-9]*
Prot	(htt|ft)
href	([ ]*href=)
URL	ps{0,1}:\/\/[a-zA-Z0-9]*\.[a-zA-Z]*
ContMeta	\"(description|keywords|author|robots)\"
UTF	\"UTF-8\"
Idioma	\"[a-z]*\"
ignora	" "|\t|\n
img	("<img ")
src	src=\"[-_a-zA-Z0-9]*(\/[-_a-zA-Z]+)*"\."[a-z]*\"|src={Enlace}

Limitaciones y consideraciones

Para el presente trabajo hubo limitaciones a la hora del desarrollo de los analizadores con respecto al lenguaje html 5. Conociendo la característica de este lenguaje y la gran cantidad de combinaciones que se pueden tener con una gran cantidad de etiquetas permitidas por html 5, se consideraron tan solo una una porción reducida de ellas (Ver apartado de gramática) y además de ello solo consideramos algunas combinaciones. El objetivo de este proyecto, no es obtener

un analizador sintáctico del lenguaje completo, sino más bien demostrar mediante los conocimientos adquiridos de la teoría que al ser aplicadas estas

Podemos experimentar con un analizador sintáctico, conocer sus reglas, sus definiciones, funcionamiento y programación del mismo. Podemos nombrar algunas de las consideraciones de nuestro analizador, algunas de ellas son:

- No contempla letras con acentos o apostrofes como (áéíó ñ etc.)
- No contempla algunos saltos de línea o una estructura del código que sea de difícil lectura para el analizador.
- Contempla un acceso por teclado del código, como la lectura de archivos de extensión .txt .html.

Ejemplo de código aceptado por nuestro analizador sintáctico:

```
<!DOCTYPE html>

<html lang="pol">

<HeAd>

        <meta charset="UTF-8">

        <meta name="description" content="Think for yourself">

        <meta name="keywords" content="Question the authory">


        <title> SINTAXIS ME DA ANSIEDAD </title>
```

```
        <meta name="author" content="SSL UTN FFRe">
</head>
<BoDy>

    <div id="intro" class="clase">
        <h1>FACULTAD</h1>
        <h2>UTN Frre</h2>
        <p> Sheets of empty canvas, untouched sheets of clay <em> Were laid spread out before me as her body once did </em> All
five </p>
        <strong> horizons revolved around her soul as the earth </strong>
        <a target="_blank" href="https://youtu.be/5ZH2it92ZmA">Pearl Jam Black</a>
        </div>
        <p> Aqui podria haber un error </p>
        <br>
        <em> I must keep reminding myself of this </em>
        <p> <em> <strong> 11:51 pm no duermo hace 3 dias </strong> </em> </p>
        <p> Vamos <em> <strong> mas <h1> profundo </h1> </strong> </em> </p>
        <p> <em> <p> <em> <p> <em> <p> VAMOS SUBIENDO </p> </em> </p> </em> </p> </em> </p>
    <section class="clase2" id="intro">
         <div>
                <p> Hace tres dias no pasaba de la primera linea </p>
                <em> Texto aqui <strong> Y... </strong> Texto alla </em>
                <H5> ALAN ES UN GRANDE ;) </h5>
         </div>
        <p> FUNCIONES TA TA TA.. </P>
   </section>
  <section id="etiquetas" >
                <p> Our body is light, we are immortal </p>
                <em> Our body is love, we are eternal </em>
                <h3>            Eternal             </h3>
```

```
                <strong>         Omniscient          </strong>

                <h5>             Omnipotent          </h5>

                <br>

                <hr>

                <br>

                <hr>

                <p>              Without judgment    </p>


  </section>


  <a href="https://developer.mozilla.org/en-US/docs/Web/HTML/Element" target="_self">World Wide Web Consortium (W3C). </a>


  <a id="wiki" class="Clase4" href="http://es.wikipedia.org:80/wiki/Special:Search?search=tren&go=Go" target="_top"> Enlace con todo </a>


  <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">

</body>

</html>
```

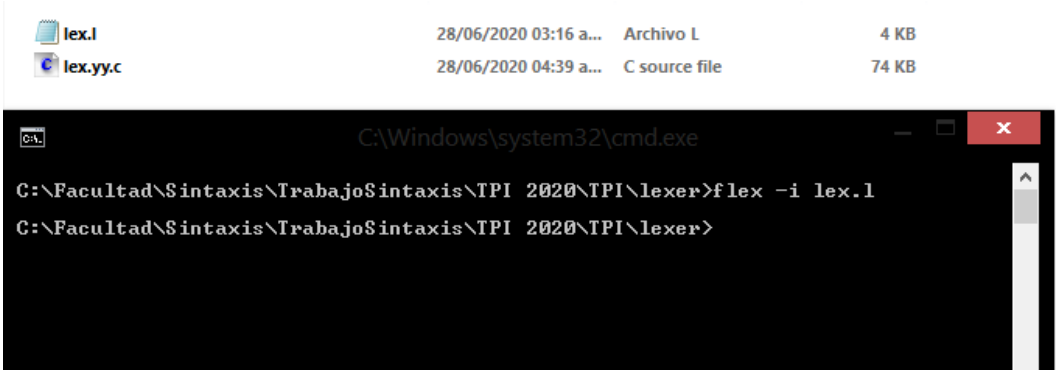
Requerimientos e instalación del analizador

Las herramientas utilizadas fueron:

- Flex versión 2.5.4
- Bison versión 2.4.1
- Sistema operativo WINDOWS 7/8/8.1/10
- GCC 9.2.0

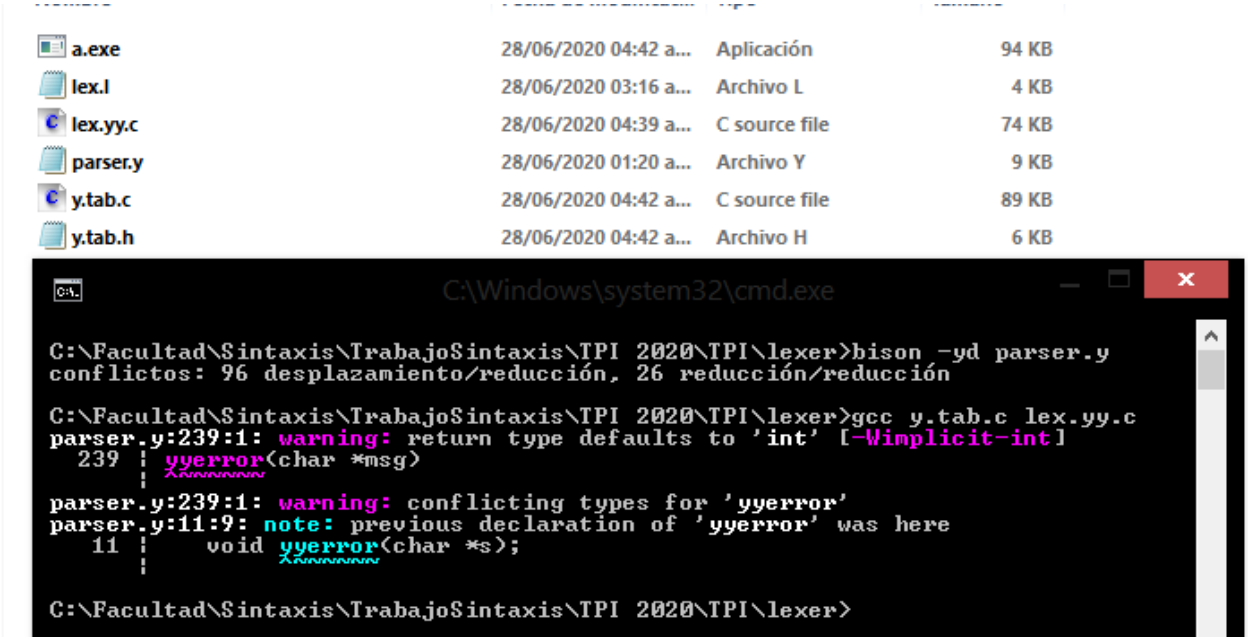
Instalación del analizador Léxico:

Primero indicamos sistema la dirección de nuestro archivo Lex.l lo que sigue es indicar en el CMD la instalación de flex, esto nos abrirá un archivo de tipo .c donde se encuentra el código fuente del lexer.



Instalación del analizador Sintáctico:

Lo siguiente será, ubicar el archivo del parser en la misma carpeta donde se aloja nuestro lexer descomprimido. Seguido a ello indicar al sistema CMD la ubicación de este archivo y descomprimirlo con bison. Se generará un archivo y.tab.c donde se ecuentra el archivo fuente del parser. Una vez hecho esto, sigue con GCC unir estos dos archivos de extensión .C, con esto obtendremos el archivo ejecutable del programa.



Conclusión:

La realización del experimento nos brindó un enorme conocimiento acerca el mundo de los compiladores, nos enriqueció en conocer las distintas herramientas y métodos para poder aplicar la teoría a la práctica. La experiencia fue de gran complejidad debido las distintas consideraciones que se tuvieron que tener y las definiciones gramaticales que se tuvieron que aplicar para lograr un programa correcto que no propague errores al momento de utilizarlo. Logramos un programa sencillo, rápido pero potente en la lectura y reconocimiento del lenguaje propuesto como experiencia el mismo es HTML 5, dicho lenguaje es ideal para poder desarrollar este tipo de proyectos debido a su gran número combinaciones, numero de etiquetas y consideraciones a ser vistas. El reconocimiento de errores y el uso de material bibliográfico fueron de gran ayuda para que este objetivo sea alcanzado. Estamos enormemente satisfechos con el programa logrado, finalizamos el mismo habiendo aplicado correctamente lo aprendido e investigado.

Bibliografía:

- ✓ Flex and Bison - John Levine
- ✓ Compiladores principios, técnicas y herramientas segunda edición – Alfred V. Aho, Monica S Lam, Ravi Sethi, Jeffrey Ullman.