Teoría de Lenguajes, autómatas y compiladores

Lenguaje para simplificar la sintaxis compleja de programación HTML que ofrece pequeños agregados de diseño con código CSS inline

2C 2022

GRUPO 2

Nombre	Apellido	Legajo	E-mail
Andres	Podgorny	60570	apodgorny@itba.edu.ar
Camila	Sierra Perez	60242	csierra@itba.edu.ar
Magdalena	Flores Lavalle	60077	magflores@itba.edu.ar
Sol Victoria	Anselmo	61278	soanselmo@itba.edu.ar

TABLA DE CONTENIDO

INTRODUCCIÓN

Nuestra idea se basa en un lenguaje para simplificar la sintaxis compleja de programación HTML y además ofrecer pequeños agregados de diseño con código CSS. Este lenguaje propone una capa de abstracción para aquellos nuevos programadores que incursionen en el mundo de la programación web pero encuentran que su sintaxis es engorrosa para iniciarse.

El objetivo de nuestro lenguaje consiste en reemplazar el uso de tags, atributos, referencias, elementos como los div y aquello que concierne a su posicionamiento y apariencia, por palabras claves (en inglés) muy simples. Así dejaríamos al usuario con una sintaxis mucho más clara para trabajar.

Para realizarlo utilizamos las tecnologías Flex y Bison.

CONSIDERACIONES ADICIONALES

Una de las consideraciones a tener en cuenta del lenguaje es que para escribir un programa vacío que sea reconocido, el mismo debe si o si tener las palabras claves START y END de la siguiente forma: sino no se generará ningún código de salida.

```
START

END
```

También, el lenguaje ignora los *whitespaces* y los atributos de cada elemento pueden estar escritos en cualquier orden, siempre y cuando luego de la palabra clave de cada atributo (a excepción de *bold*, *underlined e italics*) y antes de definir su valor se escriba el ":". Una de las cosas más importantes del lenguaje es el uso correcto de las comillas dobles (" ") para definir el contenido de los textos o las urls de los links y las imágenes. Los siguientes ejemplos generan el mismo código de salida:

```
START
FONT Arial
TITLE id:1 size:large color:red position:centered bold underlined "Hola mundo!"
TITLE id:1 bold underlined size:large position:centered color:red "Hola mundo!"
END
```

Por cómo definimos las funciones en el generator y el uso de CSS inline para ciertos tags HTML, tuvimos que definir el uso de ciertos elementos. Por ejemplo los atributos brindados ahora para los LINKS son: size, bold e italics ya que decidimos utilizar su forma default (azul y subrayado). La posición de los links, las imágenes y las tablas depende de los containers. Finalmente, para aclarar las dimensiones de las tablas se debe usar el atributo *rowxcol* y entre paréntesis su valor dividido por un "x".

```
START
FONT Cambria
TITLE id:1 “Titulo 1”
BEGIN CONTAINER position:centered
  TABLE rowxcol:(2x2)
    ROW
      TEXT size:medium position:right “Prueba 1”
      TEXT size:medium position:right “Prueba 2”
    END ROW
    ROW
      TEXT size:medium position:right “Prueba 3”
      TEXT size:medium position:right “Prueba 4”
    END ROW
  END TABLE
LINK size:small “#1”,“Volver al inicio”
IMG size:xlarge “https://imagenes.20minutos.es/google-imagenes-2.png”
END CONTAINER
END
```

DESARROLLO DEL PROYECTO

Comenzamos el proyecto realizando la entrega del Frontend, lo cual implicaba definir la gramática que iba a tener nuestro lenguaje y su objetivo, teníamos que desarrollar un lenguaje que tuviese una salida de código diferente a la ingresada y que fuera útil. Creemos que nuestro trabajo sería verdaderamente de ayuda ya que simplifica ampliamente la forma de programar páginas web simples con HTML, haciéndolo más intuitivo y simple. Esto lo logra en parte eliminando los tags, descartando la utilización de cols y rows para acomodar los elementos en la pantalla, entre otros. La simplificación la logramos realizando cambios como definir elementos de manera más sencilla, separándolos por distintas palabras clave y sin necesidad de que tenga el delimitador de cierre explícito, definir atributos de forma más directa, utilizar palabras clave más comprensibles para el usuario, como por ejemplo el uso de *title* en vez del tag *h1* o *container* en vez de *div*, entre otras.

La segunda parte del proyecto fue realizar el backend, la cual es la parte más costosa pero también ayudó tener la idea y la gramática planteadas. Sin embargo, a medida que avanzábamos, tuvimos que realizar ciertas modificaciones a nuestra gramática, tanto para simplificar el código como para poder lograr lo que queríamos. Se realizaron cambios como sacar la palabra clave de *style* para usado para el listado de los atributos *bold*, *underlined e italics*, ya que cuando tuvimos que realizar las funciones de bison, nos dimos cuenta

que estos atributos se comportan de la misma manera que los atributos *color o size*. Entre otros cambios para que el programa funcione correctamente, al estar usando CSS inline (por simplicidad de generar el código en un solo archivo), nos dimos cuenta que para ciertos tags de HTML no se podía poner exactamente la alineación en la pantalla (como a las imágenes), por lo que se les asigna en el container cuando querés posicionar alguno de esos atributos.

En resumen, nuestro lenguaje permite crear un documento HTML que contenga las estructuras principales: containers, párrafos, links, imágenes, títulos, tablas, cada uno con ciertos atributos.

Para cada entrega desarrollamos diferentes testeos que el compilador debía aceptar o rechazar, esto nos ayudaba a distinguir lo que funcionaba de lo que no y comprobar que nuestro programa estuviese andando correctamente. Una vez terminada la gramática, estos testeos sufrieron algunas modificaciones para ser acordes a la misma. Adjuntamos imágenes de los programas realizados para testear nuestro lenguaje, junto con su salida:

```
PS C:\Users\andre\Documents\TLA\TP>  .\script\test.bat
Compiler should accept...

    "R1", and it does (status 0)
    "R10", and it does (status 0)
    "R2", and it does (status 0)
    "R3", and it does (status 0)
    "R4", and it does (status 0)
    "R5", and it does (status 0)
    "R6", and it does (status 0)
    "R7", and it does (status 0)
    "R8", and it does (status 0)
    "R9", and it does (status 0)
    "RT", and it does (status 0)

Compiler should reject...

    "R11", and it does (status 1)
    "R12", and it does (status 1)
    "R13", and it does (status 1)
    "R14", and it does (status 1)
    "R15", and it does (status 1)
PS C:\Users\andre\Documents\TLA\TP>
```

- **R1**: Primer programa de prueba

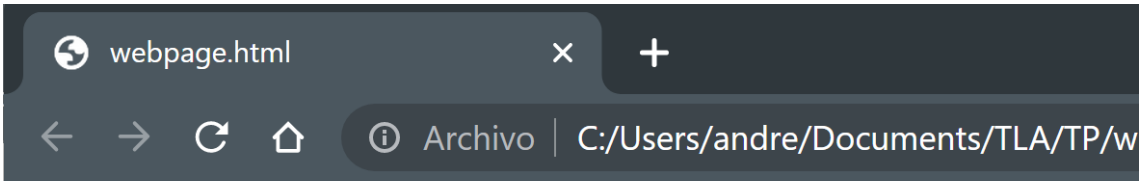
```
/* Un programa que defina una tipografía elegida por el usuario (distinta a la default). */
START

FONT Arial
TITLE "Titulo con font Arial"
TEXT "Texto con font Arial"
TEXT "Text"
TITLE "TITULO HOLA MUNDO! CON TEXTO ARIAL"
TEXT "HOLA MUNDO CON TEXTO ARIAL"

END
```

webpage.html > html > body > h3

```
<!DOCTYPE html>
<html>
    <head>
    <meta charset="utf-8">
</head>
<body>
<style>
body { font-family: 'Arial'; }</style>
<h3>Titulo con font Arial</h3>
<p>Texto con font Arial</p>
<p>Text</p>
<h3>TITULO HOLA MUNDO! CON TEXTO ARIAL</h3>
<p>HOLA MUNDO CON TEXTO ARIAL</p>
</body>
</html>
```



Titulo con font Arial

Texto con font Arial

Text

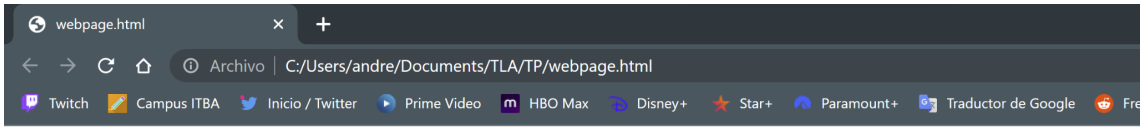
TITULO HOLA MUNDO! CON TEXTO ARIAL

HOLA MUNDO CON TEXTO ARIAL

- **R3**: Tercer programa de prueba

```
/* Un programa que produzca un elemento que contenga estilo definido, agregando CSS. */
START
FONT Arial
TITLE id:A1 size:large color:red position:center bold underlined "Hola mundo!"
TEXT size:medium position:center "Este es un mensaje de prueba"
IMG size:small "https://imagenes.20minutos.es/files/image_990_v3/uploads/imagenes/2021/11/24/google-imagenes-2.png"
LINK  size:small "https://www.google.com/","Para mas informacion haga click aqui"
LINK  size:small "#1","Volver al inicio"
END
```

```
webpage.html > html > body > h3#A1
1   <!DOCTYPE html>
2   <html>
3       <head>
4       <meta charset="utf-8">
5   </head>
6   <body>
7   <style>
8   body { font-family: 'Arial'; }</style>
9   <h3 style="color:red; text-align:center;" id="A1"><b><u>Hola mundo!</u></b></h3>
10  <p style="font-size:25pt; text-align:center;">Este es un mensaje de prueba</p>
11  <img height="200" src="https://imagenes.20minutos.es/files/image_990_v3/uploads/imagenes/2021/11/24/google-imagenes-
12  <a href="https://www.google.com/" style="font-size:14pt;">Para mas informacion haga click aqui</a>
13  <a href="#1" style="font-size:14pt;">Volver al inicio</a>
14  </body>
15  </html>
```



Hola mundo!

Este es un mensaje de prueba

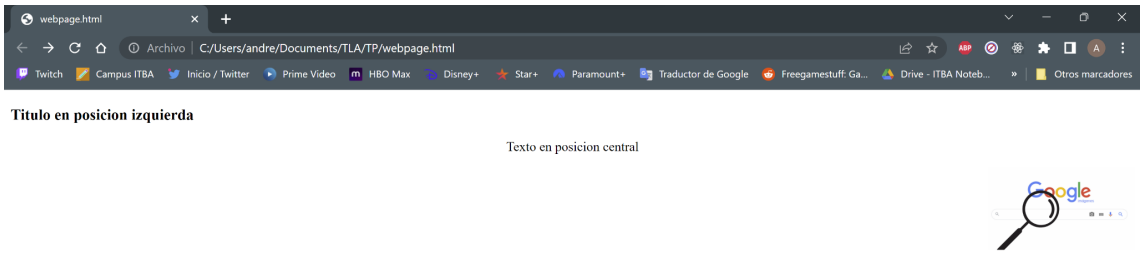
Para mas informacion haga click aqui Volver al inicio

- **R5**: Quinto programa de prueba

```
/* Un programa que genere un HTML utilizando todas las posiciones disponibles. */
START
TITLE position:left "Titulo en posicion izquierda"
TEXT position:center "Texto en posicion central"
CONTAINER position:right
IMG size:x-small "https://imagenes.20minutos.es/files/image_990_v3/uploads/imagenes/2021/11/24/google-imagenes-2.png"
END CONTAINER
END
```

webpage.html > html > body > div

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4       <meta charset="utf-8">
5   </head>
6   <body>
7   <h3 style="text-align:left;">Titulo en posicion izquierda</h3>
8   <p style="text-align:center;">Texto en posicion central</p>
9   <div style="text-align:right;">
10  <img src="https://imagenes.20minutos.es/files/image_990_v3/uploads/imagenes/2021/11/24/google-imagenes-2.png" alt="h
11  </div>
12  </body>
13  </html>
```



- **R11**: Onceavo programa de prueba

```
PS C:\Users\andre\Documents\TLA\TP>  .\script\start.bat .\test\reject\R11
[INFO ] Argumento 0: 'bin\Debug\Compiler.exe'
[INFO ] Compilando...

[DEBUG] BeginCommentPatternAction.
[DEBUG] EndCommentPatternAction.
[DEBUG] EndlinePatternAttribute: '
' (length = 1).
[DEBUG] TitlePatternTag.
[ERROR] Mensaje: 'syntax error' debido a 'TITLE' (linea 2).
[ERROR] En ASCII es:
        [84][73][84][76][69]

[ERROR] Bison finalizo debido a un error de sintaxis.
[INFO ] Fin.
PS C:\Users\andre\Documents\TLA\TP>
```

DIFICULTADES

En un principio nos fue difícil pensar en un nuevo lenguaje que pudiera ser diferente o simplificar alguno que ya exista, no se nos ocurría cómo pensarlo o la dificultad de implementarlo.

Una vez que pudimos armar la idea y empezamos con el trabajo, nos costó entender cómo funcionaban Flex y Bison; pero leyendo la documentación y consultando con el profesor Agustin pudimos comprender de a poco. Por otro lado, otro problema que tuvimos fue que queríamos realizar el código y que ya todo nos funcione de una, cuando en verdad lo que teníamos que hacer era ir probando con programas pequeños para poder detectar y arreglar los errores de a poco.

Por otro lado, al momento de armar las estructuras nos dimos cuenta de que teníamos que revisar y adaptar la gramática que habíamos planteado originalmente.

FUTURAS MODIFICACIONES

Creemos que hay múltiples modificaciones que pueden ser útiles para nuestro lenguaje, dado que HTML es muy amplio y no abarcamos todas sus componentes ni funcionalidades. Por empezar, lo siguiente a agregarle al trabajo sería la tabla de símbolos y los scopes, que no pudimos realizar por falta de tiempo para esta entrega, para agregarle la funcionalidad al lenguaje de poder reconocer, cuando se hace un hipervínculo a otro elemento, si el ID ingresado existe o no

También, sería interesante seguir ampliando el lenguaje para no solamente usar CSS inline con HTML, sino poder generar un archivo CSS para agregar características visuales complejas a la página web creada. Yendo más allá, se podría ampliar el lenguaje con ayuda de ciertas librerías como Bootstap para incluir más elementos y luego completarlo con javascript para realmente poder brindar todo el poder de creación de una página web funcional.

BIBLIOGRAFÍA

Para realizar el trabajo utilizamos toda la bibliografía provista por la cátedra:

- https://web.iitd.ac.in/~sumeet/flex__bison.pdf
- https://westes.github.io/flex/manual/Patterns.html
- https://westes.github.io/flex/manual/Start-Conditions.html
- https://gnuu.org/2009/09/18/writing-your-own-toy-compiler/
- https://www.gnu.org/software/bison/manual/html_node/Semantics.html
- https://docs.google.com/document/d/1KF-qylH4ciL5WjRQRuRSulp2xFb38X6a4THut2GDcfQ/edit#heading=h.cwvjscgwbtnh