Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Compiladores e Intérpretes

Profesor: Andrei Fuentes Leiva

Proyecto III Analizador Semántico para XHTML

Estudiantes: Daniel Cortés Sáenz, Isaac Ramírez Solano

I Semestre, 2013

Índice

Descripción del Problema

El análisis léxico (scanner) es el primer paso que se realiza en el proceso de compilación. En este se verifica que todos los caracteres y símbolos ingresados se han escrito correctamente. En el primer proyecto se implementó el analizador léxico para el lenguaje XHTML.

Para este curso se desarrollará un compilador para el lenguaje XHTML. La primera parte del proyecto constituye el desarrollo del analizador léxico para este lenguaje. XHTML es una variante de HTML pero un poco más estricto en cuanto a algunas de sus reglas. Estas reglas tuvieron que ser investigadas para determinar los tokens que retornará el scanner y que leerá el analizador sintáctico.

La especificación del analizador léxico fue desarrollada en Flex. Los errores fueron manejados por el stderr y la entrada y salida de datos se hizopor medio del stdin y stdout respectivamente.

El análisis sintáctico, es el segundo paso que se realiza en el proceso de compilación. En este paso, se verifica que se cumplan las reglas de concordancia y jerarquía del lenguaje, en este caso XHTML. Para este paso, se utilizaron reglas gramaticales que definían la estructura de los tokens permitidos en el lenguaje.

Para este tercer proyecto, se debe implementar un analizador semántico de XHTML utilizándolo en conjunto con el analizador léxico y sintáctico del primer y segundo proyecto respectivamente. Para este proyecto, no se utiliza una herramienta como Flex o Bison. Más bien, se usa lo implementado en el analizador sintáctico para definir otras reglas, que serán las que tendrán aspectos como valores de los atributos y tipos de datos permitidos.

Diseño del Programa

Decisiones de Diseño

Para la implementación del analizador semántico se usaron reglas propias. Es decir, no se utilizó ninguna herramienta como Flex o Bison. Se reutilizó el código del analizador sintáctico y se adaptó ya que en el proyecto anterior se adelantó trabajo del analizador semántico.

Algoritmos usados

Lista de Tokens

Los tokens fueron definidos dentro del archivo parser.y

Token	Descripción
ERROR	Token para identificar los errores léxicos. Este se utiliza para los caracteres inválidos.
A	Token para la etiqueta <a>
_A	Token para la etiqueta
ACRONYM	Token para la etiqueta <acronym>
_ACRONYM	Token para la etiqueta </acronym>
ADDRESS	Token para la etiqueta <address>
_ADDRESS	Token para la etiqueta </address>
APPLET	Token para la etiqueta <applet>

_APPLET	Token para la etiqueta </applet>
AREA	Token para la etiqueta <area>
_AREA	Token para la etiqueta </area>
B	Token para la etiqueta
_B	Token para la etiqueta
BASE	Token para la etiqueta <base>
_BASE	Token para la etiqueta </base>
BASEFONT	Token para la etiqueta <basefont>
_BASEFONT	Token para la etiqueta </basefont>
BDO	Token para la etiqueta <bdo>
_BDO	Token para la etiqueta </bdo>
BIG	Token para la etiqueta <big>
_BIG	Token para la etiqueta </big>
BLOCKQUOTE	Token para la etiqueta <blockquote>
_BLOCKQUOTE	Token para la etiqueta </blockquote>
BODY	Token para la etiqueta <body>
_BODY	Token para la etiqueta </body>
BR	Token para la etiqueta
_BR	Token para la etiqueta </br>
BUTTON	Token para la etiqueta <button>
_BUTTON	Token para la etiqueta </button>
CAPTION	Token para la etiqueta <caption>
_CAPTION	Token para la etiqueta </caption>
CENTER	Token para la etiqueta
_CENTER	Token para la etiqueta
CITE	Token para la etiqueta <cite>
_CITE	Token para la etiqueta </cite>
CODE	Token para la etiqueta <code>
_CODE	Token para la etiqueta </code>
COL	Token para la etiqueta <col>
_COL	Token para la etiqueta </col>
COLGROUP	Token para la etiqueta <colgroup>
_COLGROUP	Token para la etiqueta </colgroup>
DD	Token para la etiqueta <dd>
_DD	Token para la etiqueta </dd>

DEL	Token para la etiqueta <del>
_DEL	Token para la etiqueta </del>
DIR	Token para la etiqueta <dir>
_DIR	Token para la etiqueta </dir>
DIV	Token para la etiqueta
_DIV	Token para la etiqueta
DFN	Token para la etiqueta <dfn>
_DFN	Token para la etiqueta </dfn>
DL	Token para la etiqueta <dl>
_DL	Token para la etiqueta </dl>
DT	Token para la etiqueta <dt>
_DT	Token para la etiqueta </dt>
EM	Token para la etiqueta
_EM	Token para la etiqueta
FIELDSET	Token para la etiqueta <fieldset>
_FIELDSET	Token para la etiqueta </fieldset>
FONT	Token para la etiqueta
_FONT	Token para la etiqueta
FORM	Token para la etiqueta <form>
_FORM	Token para la etiqueta </form>
FRAME	Token para la etiqueta <frame>
_FRAME	Token para la etiqueta </frame>
FRAMESET	Token para la etiqueta <frameset>
_FRAMESET	Token para la etiqueta </frameset>
H1	Token para la etiqueta <h1>
_H1	Token para la etiqueta </h1>
H2	Token para la etiqueta <h2>
_H2	Token para la etiqueta </h2>
H3	Token para la etiqueta <h3>
_H3	Token para la etiqueta </h3>
H4	Token para la etiqueta <h4>
_H4	Token para la etiqueta </h4>
H5	Token para la etiqueta <h5>
_H5	Token para la etiqueta </h5>
H6	Token para la etiqueta <h6>

_H6	Token para la etiqueta </h6>
HEAD	Token para la etiqueta <head>
_HEAD	Token para la etiqueta </head>
HR	Token para la etiqueta <hr>
_HR	Token para la etiqueta </hr>
HTML	Token para la etiqueta <html>
_HTML	Token para la etiqueta </html>
I	Token para la etiqueta
_I	Token para la etiqueta
IFRAME	Token para la etiqueta <iframe>
_IFRAME	Token para la etiqueta </iframe>
IMG	Token para la etiqueta <img>
_IMG	Token para la etiqueta </img>
INPUT	Token para la etiqueta <input>
_INPUT	Token para la etiqueta </input>
INS	Token para la etiqueta <ins>
_INS	Token para la etiqueta </ins>
ISINDEX	Token para la etiqueta <isindex>
_ISINDEX	Token para la etiqueta </isindex>
KBD	Token para la etiqueta <kbd>
_KBD	Token para la etiqueta </kbd>
LABEL	Token para la etiqueta <label>
_LABEL	Token para la etiqueta </label>
LEGEND	Token para la etiqueta <legend>
_LEGEND	Token para la etiqueta </legend>
LI	Token para la etiqueta
_LI	Token para la etiqueta
LINK	Token para la etiqueta <link>
_LINK	Token para la etiqueta </link>
MAP	Token para la etiqueta <map>
_MAP	Token para la etiqueta </map>
MENU	Token para la etiqueta <menu>
_MENU	Token para la etiqueta </menu>
META	Token para la etiqueta <meta>
_META	Token para la etiqueta </meta>

NOFRAMES	Token para la etiqueta <noframes>
_NOFRAMES	Token para la etiqueta </noframes>
NOSCRIPT	Token para la etiqueta <noscript>
_NOSCRIPT	Token para la etiqueta </noscript>
OBJECT	Token para la etiqueta <object>
_OBJECT	Token para la etiqueta </object>
OL	Token para la etiqueta
_OL	Token para la etiqueta
OPTGROUP	Token para la etiqueta <optgroup>
_OPTGROUP	Token para la etiqueta </optgroup>
OPTION	Token para la etiqueta <option>
_OPTION	Token para la etiqueta </option>
P	Token para la etiqueta
_P	Token para la etiqueta
PARAM	Token para la etiqueta <param>
_PARAM	Token para la etiqueta </param>
PRE	Token para la etiqueta <pre>
_PRE	Token para la etiqueta </pre>
Q	Token para la etiqueta <q>
_Q	Token para la etiqueta </q>
S	Token para la etiqueta <s>
_S	Token para la etiqueta </s>
SAMP	Token para la etiqueta <samp>
_SAMP	Token para la etiqueta </samp>
SCRIPT	Token para la etiqueta <script>
_SCRIPT	Token para la etiqueta </script>
SELECT	Token para la etiqueta <select>
_SELECT	Token para la etiqueta </select>
SMALL	Token para la etiqueta <small>
_SMALL	Token para la etiqueta </small>
SPAN	Token para la etiqueta
_SPAN	Token para la etiqueta
STRIKE	Token para la etiqueta <strike>
_STRIKE	Token para la etiqueta </strike>
STRONG	Token para la etiqueta

_STRONG	Token para la etiqueta
STYLE	Token para la etiqueta <style>
_STYLE	Token para la etiqueta </style>
SUB	Token para la etiqueta <sub>
_SUB	Token para la etiqueta </sub>
SUP	Token para la etiqueta <sup>
_SUP	Token para la etiqueta </sup>
TABLE	Token para la etiqueta <table>
_TABLE	Token para la etiqueta </table>
TBODY	Token para la etiqueta <tbody>
_TBODY	Token para la etiqueta </tbody>
TD	Token para la etiqueta <td>
_TD	Token para la etiqueta </td>
TEXTAREA	Token para la etiqueta <textarea>
_TEXTAREA	Token para la etiqueta </textarea>
TFOOT	Token para la etiqueta <tfoot>
_TFOOT	Token para la etiqueta </tfoot>
TH	Token para la etiqueta <th>
_TH	Token para la etiqueta </th>
THEAD	Token para la etiqueta <thead>
_THEAD	Token para la etiqueta <thead>
TITLE	Token para la etiqueta <title>
_TITLE	Token para la etiqueta </title>
TR	Token para la etiqueta <tr>
_TR	Token para la etiqueta </tr>
TT	Token para la etiqueta <tt>
_TT	Token para la etiqueta </tt>
U	Token para la etiqueta <u>
_U	Token para la etiqueta </u>
UL	Token para la etiqueta
_UL	Token para la etiqueta
VAR	Token para la etiqueta <var>
_VAR	Token para la etiqueta </var>

En general los tokens son de la forma TOKEN, _TOKEN, TOKEN_. El primero es en caso de que se

abra una etiqueta (ejemplo <head>), el segundo caso es cuando se cierre una etiqueta (ejemplo: </head>), y el tercer caso es cuando se abre una etiqueta con parámetros (ejemplo: <a href=”http://www.google.co.cr”>).

Validación de la semántica

Para validar la semántica del lenguaje, se usaron reglas gramaticales para las etiquetas que tienen parámetros. En general, en para cada etiqueta se definió un token de error para cuando los parámetros ingresados no son válidos. Esto debido a que primero se implementó con un token genérico para cada etiqueta. Sin embargo, hubo problemas con Bison ya que encontraba errores de Reducción-Reducción y Reducción-Desplazamiento.

Por ejemplo, para la etiqueta <a> se tiene la siguiente expresión regular:

"<a"{A_ATTRIBUTE}+ {return A;}

Esta es en caso de que los atributos sean correctos sintáctica y semánticamente. ¿Cómo se define A_ATTRIBUTE? Se define así:

A_ATTRIBUTE (" charset=\""{ENCODING}"\"")|(" coords=\""{NUMERO}","{NUMERO}","{NUMERO}","{NUMERO}"\"")|(" href=\""{URL}"\"")|(" hreflang=\""{LANGUAGE}"\"")|(" media=\""{MEDIA_VALUE}"\"")|(" name=\""{TEXTO}"\"")|(" rel=\""{REL_VALUE}"\"")|(" rev=\""{TEXTO}"\"")|(" shape=\""{SHAPE_VALUE}"\"")|(" target=\""{TARGET_VALUE}"\"")

En donde se contemplan todos los posibles atributos que puede tener <a>. Cada uno de los identificadores en mayúscula, son expresados bajo otras reglas. Por ejemplo, URL se define de la siguiente manera:

{PROTOCOLO}{HOST}{DOMINIO}{DOMINIO_DE_PRIMER_NIVEL}({DIRECTORIO})?

Y así sucesivamente se descompone en otras reglas como la de PROTOCOLO, HOST, DOMINIO, etc.

En caso de que la expresión regular no capture la etiqueta <a> con alguno de los atributos anteriores, habría otra definida abajo para detectar el error. La expresión es la siguiente:

"<a"{ATRIBUTO}+ {yylval.string = strdup(yytext); return A_TAG_ERROR;}

Como se puede ver, se retorna al analizador un token A_TAG_ERROR. Al detectarlo, lo tomará como error y lo insertará en una lista la cual irá acumulando cada uno de los errores semánticos. Sin embargo, esto no detendrá el proceso de compilación.

En general, así funciona para cada uno de las etiquetas. Al finalizar el Parsing, se imprimirá un lista con los errores semánticos. En caso de haber errores sintácticos, sí se detiene el proceso y se imprime un único error sintáctico.

Librerías externas utilizadas

No se utilizaron librerías externas. Todas las librerías utilizadas están incluidas en C.

Análisis de Resultados

Objetivos alcanzados

En este tercer proyecto se logró alcanzar todos los objetivos. Los objetivos alcanzados fueron los siguientes:

- Se determinaron estructuras no válidas dentro del documento XHTML.
    - En caso de encontrase uno, se imprime un mensaje de error mediante el stderr indicando la fila y la columna en donde se encontró la gramática no aceptada.
    - En caso de haber errores semánticos, se insertan en una lista y se imprimen en consola al finalizar el proceso de compilación.
- La información leída es almacenada y retornada de acuerdo al token leído.
- Se creó el archivo makefile que contiene las instrucciones para correr el programa.

Objetivos no alcanzados

Debido a que se lograron alcanzar todos los objetivos del proyecto, no quedan objetivos no alcanzados en este primer proyecto.

Manual de usuario (Modificar)

Para correr el programa primero se debe abrir una consola dentro del directorio del proyecto.

A continuación escribir el comando make el cual ejecutará las instrucciones para compilar el archivo analizador.l y parser.y

Al presionar enter se mostraran los comandos que se usaron para compilar el proyecto. Se podrá comprobar que no se utilizaron librerías externas. Se generarán los archivos scanner, lex.yy.c, y.output , y.tab.c , lex.yy.c y y.tab.h.

Seguidamente se generará un .o llamado analizador. Para ejecutar el parser con algún archivo de ejemplo se deberá ejecutar ./analizador < Ejemplos/Correcto1.xhtml así para cualquier archivo dentro de esa carpeta o de extensión xhtml.

Al ejecutar la instrucción se imprimirá por consola el resultado de la ejecución. Se imprime el árbol de parsing en caso de concluir el análisis sintáctico correctamente. En otro caso, se mostrará el error encontrado, reportando la línea y columna en la que se generó el error.

Conclusión Personal

El tercer proyecto sirvió para aclarar cómo funciona un analizador semántico en conjunto con el analizador léxico y sintáctico. Quedó claro que lo que se hace primero es definir expresiones regulares que capturen los caracteres y palabras permitidas dentro del lenguaje. Se deben definir también las expresiones regulares para determinar los caracteres inválidos. Posteriormente, se deben definir los tokens que el parser utilizará en la gramática. Una vez definidos los tokens, se deberán retornar cada vez que el analizador léxico los encuentra. Se debe implementar la gramática, para determinar la estructura de cada uno de esos tokens. Finalmente, los tokens son procesados por la gramática y, en caso de errores, reportarlos y terminar el análisis.

Al finalizar la gramática, se deben definir también expresiones regulares que capturen los errores semánticos, pero que no detengan el proceso de compilación. Estos errores, se deben reportar al finalizar el proceso.

Bibliografía

Bison 2.7. (n.d.). *The GNU Operating System*. Recuperado el 2 de Junio de 2013, desde http://www.gnu.org/software/bison/manual/html_node/index.html

Deitel, H. M., & Deitel, P. J. (2004). Estructuras de datos en C. *Cómo programar en C/C++ y Java* (4a ed., p. 421). México: Pearson Educación.

Hagen, C. (28 de Julio, 2006). Better error handling using Flex and Bison. *IBM - United States*. Recuperado el 2 de Junio de 2013, desde http://www.ibm.com/developerworks/library/l-flexbison/

Introduction to XHTML. (s.f.). *W3Schools Online Web Tutorials*. Recuperado el 2 de Junio de 2013, desde http://www.w3schools.com/html/html_xhtml.asp

Levine, J. R. (2009). *Flex & bison*. Beijing: O'Reilly.

Shift/Reduce - Bison 2.7. (s.f.). *The GNU Operating System*. Recuperado el 2 de Junio de 2013, desde http://www.gnu.org/software/bison/manual/html_node/Shift_002fReduce.html#Shift_002fReduce