Práctica 4
Manejo avanzado de *Bison*

Tareas

1. Estudia la información sobre trazas en el *Capítulo 8 - Depurando Su Analizador* del manual de Bison (`bison-es-1.27.pdf`).

2. Realiza el ejercicio propuesto.

3. Elabora la memoria de la práctica y entrégala junto con los ficheros fuente según el Procedimiento de Entrega de Prácticas explicado en la Introducción a las Prácticas de la Asignatura (página 3 del guión de la Práctica 1). La fecha tope de entrega será hasta el viernes 10 de enero de 2014.

Nota: El incumplimiento de las normas de entrega se reflejará en la calificación de la memoria.

- Debes crear un directorio que contenga **exclusivamente** el fichero con la memoria en formato *PDF*, los ficheros fuente con tu código (*.l* de *Flex* y *.y* de *Bison*) y los de prueba (*.txt* de texto). No uses subdirectorios.

- En caso de que el fichero .zip resultante tenga un tamaño mayor de 512 KB deberás repetir la creación del directorio dividiéndolo en varios ficheros .zip de como máximo 512 KB cada uno, en ese caso llama a los ficheros resultantes `nipPrX1.zip`, `nipPrX2.zip`, etc.

Introducción

El objetivo de esta cuarta práctica es aplicar *Flex* y *Bison* a un ejemplo real en el que se plantea la verificación (parcial) de la sintaxis de documentos *XML*. Así, después de presentar los aspectos básicos del estándar *XML*, se plantea la construcción de un analizador sintáctico que reconozca un subconjunto determinado de documentos *XML* bien formados.

El Estándar *XML*

XML es un estándar para la creación de documentos de texto con una estructura bien definida. Aunque originalmente fue concebido como un formato de intercambio de datos entre distintas plataformas, en la actualidad su uso se ha generalizado como formato nativo para ficheros en multitud de aplicaciones, para la descripción de servicios *Web*, como formato de almacenamiento en algunas bases datos, etc. El estándar es accesible desde http://www.w3.org/XML/.

Estructura de un documento *XML*

Los documentos *XML* tienen una estructura definida con elementos delimitadores (etiquetas) que admiten cierto anidamiento. Las etiquetas *XML* son, básicamente, un texto entre los símbolos '<' y '>'. En *XML* se distinguen las mayúsculas de las minúsculas, con lo que $< ETIQUETA >$ y $< etiqueta >$ son etiquetas distintas. Hay etiquetas de apertura y etiquetas de cierre:

$< ETIQUETA > Contenido\ de\ etiqueta < /ETIQUETA >$

Puede haber etiquetas que aparezcan sin etiqueta de cierre (elemento vacío), pero lleven una barra '/' al final:

$< ETIQUETA/ >$

Los elementos pueden tener atributos en sus etiquetas de apertura o en su única etiqueta si son elementos vacíos. Los atributos tienen sus valores entre comillas (simples o dobles) y, si hay varios, tienen que ir separados por al menos un espacio (tabulador, fin de línea...).

$< ETIQUETA\ ATRIBUTO =$"$Valor$"$\quad ATRIBUTO2 = 'Valor' >$

Los comentarios se escriben de la siguiente forma y pueden ocupar varias líneas:

<! − − Esto es un comentario en XML −− >

XML bien formado

XML tiene una serie de reglas sintácticas determinadas:

- Los documentos *XML* comienzan con una declaración de la versión del estándar *XML* que cumple. Esta declaración puede llevar otros atributos opcionales:
  $<?xml\ version =$"1.0"?>

- El documento tiene un solo elemento raíz que engloba a todos los demás entre una etiqueta de apertura y una de cierre. Los elementos englobados por otro se denominan sub-elementos o hijos.

- Todas las etiquetas de apertura deben tener su correspondiente etiqueta de cierre para los elementos que contengan a otros, o que contengan datos. Si aparece una etiqueta sola es un elemento vacío y debe tener una barra '/' al final.

- Entre dos etiquetas puede haber espacios, texto y/o otras etiquetas. También puede no haber nada entre dos etiquetas.

- No pueden aparecer dentro del texto de un elemento caracteres especiales como es el caso de '<' (en su lugar se escribe &*lt*;)

- Los elementos deben anidarse correctamente. Esto quiere decir que la etiqueta de cierre de un elemento hijo de otro debe aparecer antes que la etiqueta de cierre de su elemento padre. Si aparecen atributos en algún elemento tienen que tener un valor escrito entre comillas simples o dobles.

Un ejemplo de documento XML bien formado:

<?*xml version* ="1.0" encoding= "ISO-8859-1"?>
<Biblioteca>
<Libro>
La catalogación de este libro se ha llevado a cabo a partir
de información histórica recopilada de varias fuentes
<Titulo>
El ingenioso hidalgo Don Quijote de la Mancha
</Titulo>
<Autor>
<Nombre tipo = "Nombre">
Miguel de Cervantes
</Nombre>
<Nombre tipo = "Apodo">
El Manco de Lepanto
</Nombre>
</Autor>
<Novela tipoNovela = "Caballerías"/>
</Libro>
<Libro>
<Titulo>
Introducción a la Teoría de Autómatas, Lenguajes y Computación
</Titulo>

```
    <Autor>
      <Nombre tipo = "Nombre">
        John E. Hopcroft
      </Nombre>
    </Autor>
    <LibroTexto clasificación = "Informática Teórica"/>
  </Libro>
</Biblioteca>
```

XML válido

Un documento *XML* válido es aquel que además de estar bien formado, es conforme a cierta estructura previamente establecida. Esta estructura se especifica en forma de *Definición de Tipo de Documento* (*DTD*) o mediante un *esquema* (*Schema*). Un *DTD* o un esquema son gramáticas que especifican qué elementos pueden y/o deben aparecer en un documento *XML* y cómo deben estar estructurados. Los esquemas son más potentes que los *DTDs* y permiten definir con precisión los tipos de datos válidos en elementos y atributos, cardinalidades complejas, etc.... **Nótese que en esta práctica no se aborda en ningún momento la validez de documentos *XML*.**

Lectura de *XML* con *Flex*

A continuación se proporciona un esqueleto del programa en *Flex* que se usará para el análisis de ficheros *XML*. Recoge los patrones a reconocer y se proporciona un ejemplo comentado. La primera tarea de la práctica será completarlo:

```
%{
include "y.tab.h"
%}
/* ...DEFINICIONES... */
%%
{openTag}          {return(OPEN_TAG);      /* "< HOLA a ='45'>" */}
{closeTag}         {return(CLOSE_TAG);     /* "< /HOLA >" */}
{emptyTag}         {return(EMPTY_TAG);     /* "< HOLA A ='12'/ >" */}
{instr}            {return(INSTR);     /* "<?xml version ="1.0"? > " */}
{comment}          {/* LOS IGNORO */ /* "<! -- COM --  >" */}
{spaces}           {/* LOS IGNORO */ /* "/n/t " */ }
{cdata}            {return(CDATA);    /* "Text = 3, &vale casi todo.\t\n" */ }
.                  {return(BADCAR); /* "<" */}
%%
```

Notas:

- Esta no es, ni mucho menos, la única manera de analizar léxicamente la entrada. Es una propuesta bastante completa y no muy compleja.
- Si una etiqueta lleva varios atributos, éstos deberán estar separados por espacios.
- Se ignoran comentarios y espacios, salvo los que van entre los atributos. Éstos los trataremos dentro del patrón de la etiqueta correspondiente, dado que en general podemos organizar las etiquetas en un documento con margen para usar tabulaciones, espacios y fines de línea para indentarlo. Los comentarios se podrán poner allí donde se requieran.
- El patrón *CDATA* empareja el texto libre que podemos escribir entre etiquetas. Se admite casi cualquier carácter (incluyendo espacios, saltos de línea, signos de puntuación...), salvo el de abrir etiqueta '<' y algún otro del mismo tipo.
- Se tratan de forma distinta espacios y *CDATA* porque este último debe aparecer en nuestra gramática. Así, hay sitios donde es aceptable y sitios donde no lo es. En cuanto a los espacios, se podrán ignorar y no ser devueltos a *Bison,* excepto aquellos que estén entre atributos.
- Se devolverá un token cuando se encuentre carácter que no se ha emparejado antes. Casi con seguridad será un error del fichero *XML* y de esta forma es más sencillo que *Bison*. Dicho token no aparecerá en la gramática de *Bison*.

Análisis sintáctico con *Bison*

Construye una gramática en Bison que acepte ficheros *XML* bien formados. Los tokens se corresponderán con aquello que devuelva *Flex* y están en el esqueleto que se facilita en el ejemplo.

Notas:

- Los ficheros *XML* a aceptar tendrán la siguiente forma:
  $INSTR$
  $OPEN\_TAG$
  $RESTO\ DEL\ DOCUMENTO$ ($CDATA$, $ELEMENTOS\ ANIDADOS$, $ELEMENTOS\ VACIOS \ldots$)
  $CLOSE\_TAG$

  Se requiere que el documento empiece con una instrucción *XML*, que tenga una sola etiqueta de primer nivel (que englobe a todas las demás), y que tenga cualquier contenido aceptable (*CDATA* y elementos correctamente anidados).

- No se puede comprobar si la etiqueta de inicio corresponde con la de cierre sin entrar en aspectos más complejos de *Bison*, así que se aceptarán como buenos documentos incorrectos como este:
  $<?xml\ version =$"1.0"?>
  $< abro >$
  $< /cierro >$

- Para hacer trazas de la ejecución de *Bison* se pueden utilizar acciones dentro de las reglas, con la misma estructura que ya se explicó para *Flex*. Aunque estas acciones sirven para muchas más cosas, se pueden utilizara para localizar problemas en la gramática. Por ejemplo:
  $S :\ \{printf($ "$Entra\ en\ ELEMENTOS \backslash n$"); $ELEMENTOS\ \{printf($"$Entra\ en\ FIN \backslash n$"); FIN ;
  Esto escribirá por pantalla una traza del análisis realizado por la gramática.

- En la página web de la asignatura hay un paquete de ficheros *XML* para realizar pruebas sencillas. Adicionalmente, se valorarán muy positivamente las pruebas con otros ficheros que se incluyan.