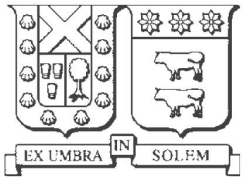
Departamento de Informática
Universidad Técnica Federico Santa María



Tarea N°4
Estructura de Datos
(8) XML's style (8)

Hubert Hoffmann Nagel
hoffmann@inf.utfsm.cl

Ariel Sanhueza Román
asanhuez@alumnos.inf.utfsm.cl

Vicente Lizana Estivill
vlizana@alumnos.inf.utfsm.cl

12 de mayo de 2015

1. Objetivos

- Que los estudiantes implementen el TDA Árbol.
- Que los estudiantes sepan utilizar los TDA ya aprendidos.
- Que los estudiantes trabajen en equipo para la resolución de la tarea.

2. Instrucciones

El XML (eXtensible Markup Language) es un lenguaje de marcado que permite estructurar información de manera simple. Es muy similar en estructura al HTML. Un ejemplo del contenido de un archivo xml es el siguiente:

```
<?xml version="1.0" encoding="UTF-8" ?>
<listado_curso>
  <estudiante>
    <nombre>
      Arthas Menethil
    </nombre>
    <ciudad>
      Lordaeron
    </ciudad>
    <rol>
      201073034-5
    </rol>
  </estudiante>
  <estudiante>
    <nombre>
      Thrall
    </nombre>
```

```
    <ciudad>
      Orgrimmar
    </ciudad
    <rol>
      201173003-k
    </rol>
  </estudiante>
</listado_curso>
```

Su tarea será, mediante el uso de árboles n-arios, la implementación de un lector de archivos XML "simplificados" (desde ahora llamado XMLS: XML Simplificado).

Su programa deberá leer un archivo en formato XMLS (se especifica en la siguiente sección) y guardarlo en un árbol n-ario que se utilizará para guardar la jerarquía de los *tags*.

3. Archivo XMLS

Para trabajar la tarea usaremos un archivo XML simplificado. Originalmente, uno puede agregar propiedades a cada tag. Para esta tarea no habrán atributos en los tags. Esto quiere decir que un tag simplemente contiene el nombre de este. Además, cada línea del archivo será o un tag o un string con el dato contenido en el tag. Para facilitar aún más la lectura de este, el contenido de los tags no tendrá espacios y será de largo máximo 50. Además, nuestros XMLS no contendrán la línea inicial que indica que el archivo es XML (línea inical del ejemplo).

4. Entrada y salida

El programa recibirá inicialmente un string, el cual será el nombre del archivo XMLS. Luego de esto su programa debería imprimir el mensaje **Cargando archivo.**. Cuando su programa termine de guardar el archivo imprimirá el mensaje **Archivo cargado.**.

Luego de haber realizado lo anterior, el programa nos llevará a una consola, la cual nos permitirá ir navegando entre los distintos tags leídos desde el archivo. Inicialmente el tag de inicio es el tag *root* (el que abarca todo el contenido, en el ejemplo anterior era el tag listado_curso). Cuando la consola esté esperando un comando, se deberá imprimir **>** (caracter >y un espacio). Nuestra consola aceptará solo 5 comandos:

- **list**: Lista todos los tags contenidos en el tag actual
- **change x**: Donde **x** es un número mayor o igual a 1, indicando el tag al cual queremos navegar.
- **back**: Ir al tag anterior en la jerarquía.
- **position**: Indica el tag en el cual estamos situados actualmente.
- **exit**: Salir del progrma.

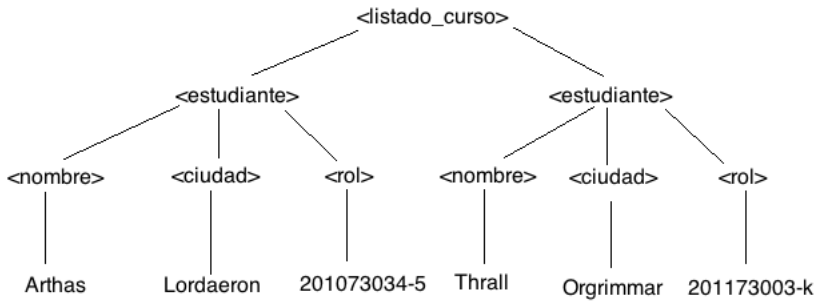
5. Ejemplo de entrada y salida de datos.

Un ejemplo de archivo XMLS válido sería:

```
<listado_curso>
  <estudiante>
    <nombre>
      Arthas
    </nombre>
    <ciudad>
      Lordaeron
    </ciudad>
    <rol>
      201073034-5
```

```
    </rol>
  </estudiante>
  <estudiante>
    <nombre>
      Thrall
    </nombre>
    <ciudad>
      Orgrimmar
    </ciudad
    <rol>
      201173003-k
    </rol>
  </estudiante>
</listado_curso>
```

Un archivo como el anterior debería generar un árbol como el siguiente:



Un ejemplo de E/S de la tarea sería:

```
$ ./tarea4
listado_curso.xmls
Cargando archivo.
Archivo cargado.
> position
<listado_curso>
> list
<estudiante>
<estudiante>
> change 1
> list
<nombre>
<ciudad>
<rol>
> change 1
> position
<nombre>
> list
Arthas
> back
> change 2
> position
<ciudad>
> list
Lordaeron
```

```
> back
> position
<estudiante>
> exit
$
```

6. Consideraciones adicionales.

- Todas las entradas ingresadas son correctas.
- Se permite y se **recomienda** el uso de la STl. Se permitirá el uso de las bibliotecas: list, stack, queue, vector, array, algorithm.
- Se repite: los tags y contenidos de este no tendrán espacios.
- **NO** es un supuesto válido un número fijo de hijos máximos (de etiquetas dentro de un bloque en el archivo XMLS).

7. Bonificaciones adicionales.

- No hay bonificaciones adicionales en esta tarea.

8. Evaluación del código.

- Implementación del árbol n-ario: 80 pts.
- Uso del árbol para la solución del problema. 20 pts.

9. Sobre entrega.

- La fecha límite de entrega de la tarea es el día lunes 25 de mayo antes de las 23:55 hrs.
- Para despejar dudas sobre la tarea o el reglamento de tareas puede consultar en la plataforma Moodle en la sección correspondiente.