

# Librería XML

Curso 2004/05, Fecha:28/10/2004

## Enunciado 1 (HTML) *Se pide:*

- Almacenar el siguiente código HTML en un fichero llamado *Ejemplo.html* y visualizarlo en un navegador Web.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Ejemplo</title>
</head>
<body>
<h1>Ejemplo de HTML</h1>
<p>Esto es un texto...</p>
</body>
</html>
```

- Modificar el fichero anterior para que incluya un enlace a la página de la asignatura. En HTML, los enlaces se incluyen mediante el formato

```
<a href="http://URL_a_enlazar">
texto
</a>
```

## Enunciado 2 (SVG) *Se pide:*

- Almacenar el siguiente código en un fichero llamado *Dibujo.svg*

```
<svg xmlns="http://www.w3.org/2000/svg">
<circle cx="100" cy="100" r="50" fill="green"/>
<circle cx="100" cy="100" r="30" fill="red"/>
</svg>
```

- Modificar el dibujo para que incluya un rectángulo. En SVG, los rectángulos se incluyen mediante el formato

```
<rect x=".." y=".."
width=".." height=".." />
```

**Enunciado 3 (Objeto)** La instrucción `object` de HTML permite insertar un objeto dentro de una página Web. Por ejemplo, para insertar un dibujo SVG almacenado en un fichero `dibujo.svg` puede utilizarse:

```
<object data="dibujo.svg"
        type="image/svg+xml">
    Formato SVG no visible
</object>
```

Realizar varios dibujos SVG e insertarlos dentro de una página Web para formar una especie de galería

**Enunciado 4 (Compilar)** Cargar el siguiente programa en un entorno Haskell

```
> module LibXML(gen, genAs, vacio) where
> dibujo = "<svg_xmlns=\http://www.w3.org/2000/svg\>"
>         ++ contenido ++ "</svg>"
>
> contenido = "<circle_cx=\100\ _cy=\100\ " ++
>             "_r=\50\ _fill=\green\>"
>             ++ "<circle_cx=\100\ _cy=\100\ " ++
>             "_r=\30\ _fill=\red\>"
>
> main = writeFile "dibu.svg" dibujo
>
> -- vacio = undefined
> -- gen = undefined
> -- genAs = undefined
```

**Enunciado 5 (vacío)** Definir la función `vacio` que construye un elemento vacío a partir de una lista de atributos

```
> vacio :: String -> [(String, String)] -> String
> vacio e as = "<" ++ e ++ ponAtts as ++ ">"
>
> ponAtts :: [(String, String)] -> String
> ponAtts = concat . map f
>     where f (v, n) = "_" ++ v ++ "=" ++ n ++ "\""
```

Se utilizan las funciones predefinidas

- `map` ::  $(a \rightarrow b) \rightarrow [a] \rightarrow [b]$   
`map f ls` devuelve la lista que resulta de aplicar `f` a cada elemento de `ls`
- `concat` ::  $[[a]] \rightarrow [a]$   
`concat ls` devuelve la lista resultante de concatenar las listas `ls`

**Enunciado 6 (Reescribir)** *Re-escribir los elementos vacíos del programa Haskell utilizando la función vacío*

**Enunciado 7 (genAs)** *Escribir una función genAs que a partir de una etiqueta e, una lista de atributos de la forma [(a1,v1),..., (aN,vN)] y un contenido contenido, genere un elemento de la forma: <e a1="v1"... aN="vN">contenido</e> Tipo:*

```
> genAs :: String -> [(String, String)] -> String -> String
```

```
?- genAs "a" [( "href", "http://www.uniovi.es" )] "Universidad"
<a href="http://www.uniovi.es">Universidad</a>
```

**Enunciado 8 (gen)** *Basándose en la función anterior, construir una función gen que tome una etiqueta e y una cadena contenido, genere un elemento sin atributos de la forma: <e>contenido</e> Tipo:*

```
> gen :: String -> String -> String
```

```
?- gen "h1" "Dibujos"
<h1>Dibujos</h1>
```

**Enunciado 9 (Generador)** *Escribir programas Haskell utilizando las funciones anteriores que generen páginas HTML y/o ficheros SVG*

**Enunciado 10 (Círculos(Opcional))** *Se pide:*

- Construir una función que tome como parámetros las coordenadas (x,y) del centro y el radio y genere un elemento SVG representando a un círculo.

*Para convertir de números a cadenas puede utilizarse la función predefinida show*

- Construir filas, cruces o diagonales como las de las siguientes imágenes:



