

# Generación de Páginas HTML

Curso 2002/03, Fecha:11/10/2002

**Enunciado 1 (Compilar)** *Cargar el siguiente programa en un entorno Haskell*

```
> module Pf202(gen, genAs, vacio) where

> pagina = "<html>" ++ cabecera ++ cuerpo ++ "</html>"

> cabecera = "<head><title>Página</title></head>\n"
> cuerpo = "<body>" ++ info ++ "</body>\n"
> info = "<h1>Dibujos</h1><p>Ejemplo</p>" ++ objeto

> objeto = "<object_data=\dibu.svg\"_\" ++
>         \"type=\image/svg+xml\">\" ++
>         \"no_visible</object>\"

> dibujo = "<svg>" ++ cont ++ "</svg>"
> cont = "<circle_cx=\100\"_cy=\100\" ++
>        \"_r=\50\"_fill=\green\"/>\"
>        ++ "<circle_cx=\100\"_cy=\100\" ++
>        \"_r=\30\"_fill=\red\"/>\"

> main = do
>         writeFile "test.htm" pagina
>         writeFile "dibu.svg" dibujo
```

**Enunciado 2 (enlace)** *Modificar el programa anterior para que en la página Web aparezca un enlace a alguna página Web externa. En HTML, los enlaces se incluyen mediante el formato <a href="http://página\_a\_enlazar">texto</a>*

**Enunciado 3 (cuadro)** *Modificar el programa anterior para que en el dibujo se incluya un cuadro En SVG, los cuadros se incluyen mediante el formato <rect x=".." y=".." width=".." height=".." />*

**Enunciado 4 (vacío)** *Añadir la función vacío 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 → b) → [a] → [b]  
**map** f ls devuelve la lista que resulta de aplicar f a cada elemento de ls
- **concat** :: [[a]] → [a]  
**concat** ls devuelve la lista resultante de concatenar las listas ls

**Enunciado 5 (Reescribir)** Re-escribir los elementos vacíos utilizando la función vacío

**Enunciado 6 (genAs)** Escribir una función genAs que tome una etiqueta, una lista de atributos y una cadena y genere el elemento correspondiente. Tipo:

```

> genAs :: String → [(String, String)] → String → String
?–genAs "object" ["data", "dibu"] "No"
<object data="dibu">No</object>

```

**Enunciado 7 (gen)** Basándose en la función anterior, construir una función gen que tome una etiqueta y una cadena y genere el elemento correspondiente (sin atributos). Tipo:

```

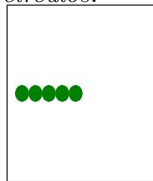
> gen :: String → String → String
?–gen "h1" "Dibujos"
<h1>Dibujos</h1>

```

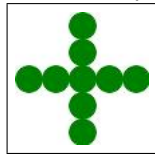
**Enunciado 8 (Reescribir2)** Re-escribir todo el programa utilizando las funciones vacío, genAs y gen

**Enunciado 9 (circulo)** 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 pre-definida show

**Enunciado 10 (fila)** Modificar el programa anterior para que genere una fila de círculos.



**Enunciado 11 (cruz(Opcional))** *Modificar el programa anterior para que genere una cruz formada por círculos.*



**Enunciado 12 (diag(Opcional))** *Modificar el programa anterior para que genere una diagonal formada por círculos.*

