

# Tipos de datos recursivos: Quadrees

Curso 2002/03, Fecha:31/10/2003

**Enunciado 1 (Quadtree)** *El siguiente programa define un tipo de datos recursivo que representa quadrees.*

```
> module Pf402 where
> import Pf202(vacio , gen , genAs)

> data Color = RGB Int Int Int
>   deriving Show

> data QT = B Color
>         | D QT QT QT QT
>   deriving Show

> rojo , verde , ej :: QT
> rojo  = B (RGB 255 0 0)
> verde = B (RGB 0 255 0)
> ej    = D rojo verde verde rojo
```

*Compilar el programa y definir algunos ejemplos de quadrees. Chequear el tipo de los ejemplos definidos.*

**Enunciado 2 (ver)** *Añadir al programa anterior el siguiente fragmento que incluye la función verqt que permite visualizar una representación del quadtree en formato SVG*

```
> type Punto = (Int, Int)
> type Dim = (Punto, Int)

> verqt :: QT → IO ()
> verqt q = writeFile "qtree.svg"
>           (gen "svg" (ver ((0,0),500) q))

> ver :: Dim → QT → String
> ver ((x,y),d) (B c) =
>   rect (x,y) (x+d+1,y+d+1) c

> ver ((x,y),d) (D ul ur dl dr) =
>   let d2 = d `div` 2
>   in
```

```

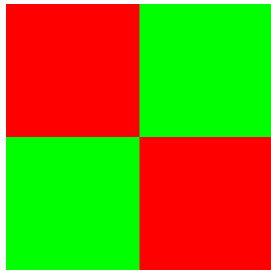
>   if d <= 0 then ""
>   else ver ((x,y),d2)      ul ++
>         ver ((x+d2,y),d2)  ur ++
>         ver ((x,y+d2),d2)  dl ++
>         ver ((x+d2,y+d2),d2) dr

> rect :: Punto → Punto → Color → String
> rect (x,y) (x',y') (RGB r g b) =
>   vacio "rect" [( "x",show x),("y",show y),
>                  ("height",show (abs (x - x'))),
>                  ("width",show (abs (y - y'))),
>                  ("fill", "rgb ("++ show r ++", "++
>                                     show g ++", "++
>                                     show b ++")"),
>                  ("stroke-width", "0")]

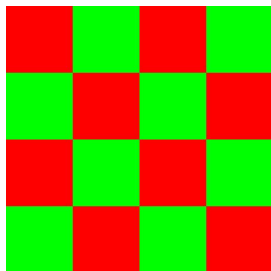
```

---

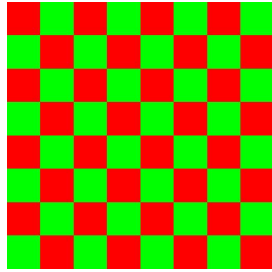
**Enunciado 3 (diag)** Construir un quadtree que represente una diagonal.



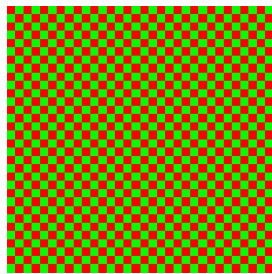
**Enunciado 4 (repite)** Construir una función *repite* que tome como argumento un quadtree y genere un nuevo quadtree repitiendo en cada cuadrante el quadtree original. Por ejemplo, al aplicar *repite* *diag* se obtendría:



**Enunciado 5 (repiteN)** Construir una función *repiteN* que tome como argumentos un número *n* y un quadtree *q* y genere un quadtree repitiendo *n* veces el quadtree *q*. Por ejemplo, para *n* = 3, se obtendrá la imagen:



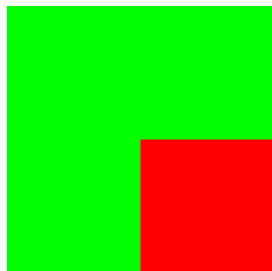
mientras que para  $n = 5$ , se obtendrá



**Enunciado 6 (rota)** Construir una función que tome un quadtree y lo rote 90 grados

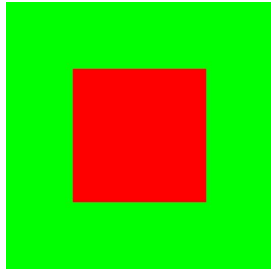
**Enunciado 7 (esquina)** Construir un programa que tome como parámetro un quadtree  $q$  y genere un nuevo quadtree con todos los cuadrantes de color verde salvo la esquina inferior derecha que tomará el valor  $q$

Al dibujar, por ejemplo, esquina rojo se obtendría

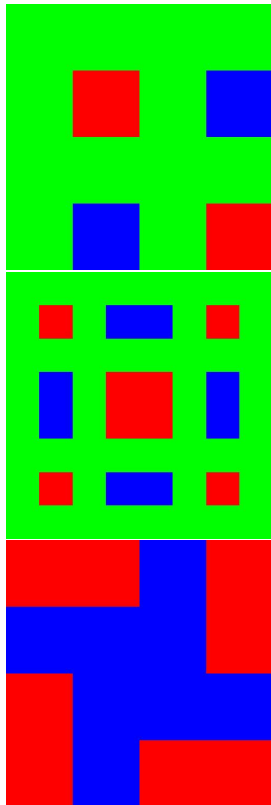


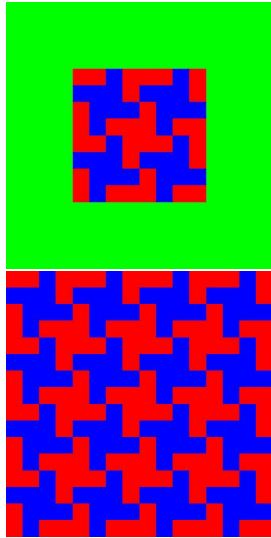
**Enunciado 8 (rotaciones)** Construir un programa que tome un quadtree y genere un nuevo quadtree a partir del anterior incluyendo en cada cuadrante una rotación de 90 grados del quadtree original

Al dibujar, por ejemplo, rotaciones (esquina rojo) se obtendría



**Enunciado 9 (otros(Opcional))** *Construir otras figuras geométricas mediante combinaciones de rotaciones y esquinas*  
*Ejemplos:*





**Enunciado 10 (librería(Opcional))** *Construir una librería de funciones que permitan manipular quadtrees para crear figuras geométricas diversas*