GTree (Ejemplos)

A continuación se muestra una ejemplo de uso del módulo GTree en una sesión con el compilador interactivo ocaml

```
~ % ocaml
OCaml version 5.2.0
Enter #help;; for help.

# #load_rec "gTree.cmo";;
# open GTree;;
# let leaf x = GT (x, []);;
val leaf : 'a -> 'a GTree.gtree = <fun>
# let t5 = GT (5, [leaf 12]);;
val t5 : int GTree.gtree = GT (5, [GT (12, [])])
# let t3 = GT (3, [t5; leaf 6; leaf 7]);;
val t3 : int GTree.gtree =
  GT (3, [GT (5, [GT (12, [])]); GT (6, []); GT (7, [])])
# let t4 = GT (4, [leaf 8; leaf 9; leaf 10; leaf 11]);;
val t4 : int GTree.gtree =
  GT (4, [GT (8, []); GT (9, []); GT (10, []); GT (11, [])])
# let t2 = GT (2, [t3; t4]);;
val t2 : int GTree.gtree =
  GT (2,
   [GT (3, [GT (5, [GT (12, [])]); GT (6, []); GT (7, [])]);
    GT (4, [GT (8, []); GT (9, []); GT (10, []); GT (11, [])])])
# let t1 = GT (1, [t2]);;
val t1 : int GTree.gtree =
  GT (1,
   [GT (2,
     [GT (3, [GT (5, [GT (12, [])]); GT (6, []); GT (7, [])]);
      GT (4, [GT (8, []); GT (9, []); GT (10, []); GT (11, [])])])])
# size t1, height t1;;
- : int * int = (12, 5)
# breadth t1;;
- : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12]
# let t1' = mirror t1;;
val t1' : int GTree.t =
  GT (1,
   [GT (2,
     [GT (4, [GT (11, []); GT (10, []); GT (9, []); GT (8, [])]);
      GT (3, [GT (7, []); GT (6, []); GT (5, [GT (12, [])])])])])
# breadth t1';;
- : int list = [1; 2; 4; 3; 11; 10; 9; 8; 7; 6; 5; 12]
# preorder t1;;
- : int list = [1; 2; 3; 5; 12; 6; 7; 4; 8; 9; 10; 11]
# postorder t1;;
- : int list = [12; 5; 6; 7; 3; 8; 9; 10; 11; 4; 2; 1]
```

```
# leaves t1;;
- : int list = [12; 6; 7; 8; 9; 10; 11]
# leaves t1';;
- : int list = [11; 10; 9; 8; 7; 6; 12]
# find_in_depth ((<) 3) t1;;
- : int = 5
# find_in_depth ((<) 3) t1';;
- : int = 4
# find_in_depth_opt ((<) 11) t1';;
- : int option = Some 12
# find_in_depth_opt ((<) 12) t1';;
- : int option = None
# breadth_find ((<) 4) t1;;
- : int = 5
# breadth_find ((<) 4) t1';;
- : int = 11
# breadth_find_opt ((<) 11) t1';;
- : int option = Some 12
# breadth_find_opt ((<) 12) t1';;
- : int option = None
# exists ((<) 11) t1;;
- : bool = true
# exists ((<) 12) t1;;
- : bool = false
# for_all ((<) 7) t1;;
- : bool = false
# for_all ((<) 0) t1;;
- : bool = true
# for_all ((<) 4) t1;;
- : bool = false
# for_all ((<) 4) t5;;
- : bool = true
# let t1c = map (fun n -> char_of_int (96 + n)) t1;;
val t1c : char GTree.t =
  GT ('a',
   [GT ('b',
     [GT ('c', [GT ('e', [GT ('l', [])]); GT ('f', []); GT ('g', [])]);
      GT ('d', [GT ('h', []); GT ('i', []); GT ('j', []); GT ('k', [])])])])
# breadth t1c;;
- : char list = ['a'; 'b'; 'c'; 'd'; 'e'; 'f'; 'g'; 'h'; 'i'; 'j'; 'k'; 'l']
# let t = replace_when (fun i -> i mod 3 = 0) t1 (GT (0, [leaf (-1)]));;
val t : int GTree.t =
  GT (1,
   [GT (2,
     [GT (0, [GT (-1, [])]);
      GT (4, [GT (8, []); GT (0, [GT (-1, [])]); GT (10, []); GT (11, [])])])])
# breadth t;;
- : int list = [1; 2; 0; 4; -1; 8; 0; 10; 11; -1]
# leaves t;;
- : int list = [-1; 8; -1; 10; 11]
```

```
# let ta = cut_below (fun i -> i mod 3 = 0) t1;;
val ta : int GTree.t =
  GT (1,
   [GT (2,
     [GT (3, []); GT (4, [GT (8, []); GT (9, []); GT (10, []); GT (11, [])])])])
# breadth ta;;
- : int list = [1; 2; 3; 4; 8; 9; 10; 11]
# let stl = StBinTree.leaftree;;
val stl : 'a -> 'a StBinTree.t = <fun>
# let comb = StBinTree.comb;;
val comb : 'a -> 'a StBinTree.t -> 'a StBinTree.t -> 'a StBinTree.t = <fun>
# let s1 = comb 1 (stl 2) (stl 3);;
val s1 : int StBinTree.t =
  StBinTree.SBT (StBinTree.Leaf 2, 1, StBinTree.Leaf 3)
# let s4 = comb 4 (stl 5) (stl 6);;
val s4 : int StBinTree.t =
  StBinTree.SBT (StBinTree.Leaf 5, 4, StBinTree.Leaf 6)
# let s = comb 0 s1 s4;;
val s : int StBinTree.t =
  StBinTree.SBT (StBinTree.SBT (StBinTree.Leaf 2, 1, StBinTree.Leaf 3), 0,
   StBinTree.SBT (StBinTree.Leaf 5, 4, StBinTree.Leaf 6))
# breadth (from_bin (StBinTree.to_bin s));;
- : int list = [0; 1; 4; 2; 3; 5; 6]
# breadth (from_stbin s);;
- : int list = [0; 1; 4; 2; 3; 5; 6]
# #quit;;
~ %
```

Aunque es interesante que le eche un ojo a toda las frases de este ejemplo, si simplemente quiere asegurarse de que con su módulo BinTree genera la misma salida, puede probar lo siguiente

```
~ % ocaml -no-version -noprompt binTree.cmo stBinTree.cmo gTree.cmo < ex3.ml > out
~ % diff out ex3.out
~ %
```