PP-arboles.pdf

Dieeelg

Paradigmas de Programación

2º Grado en Ingeniería Informática

Facultad de Informática
Universidad de A Coruña

PP ÁRBOLES

Definicion de tipo de arbol:

```
type 'a sttree =
 S of 'a
|C of 'a * 'a sttree * 'a sttree;;
type 'a tree = S of 'a | C of 'a * 'a tree * 'a tree
```

```
S 3;;
- : int tree = S 3
```

```
S 'a';;
- : char tree = S 'a'
```

```
S 3 = S 5;;
- : bool = false
```

```
S 3 = 3;;
Esta expresión tiene tipo int pero se esperaba una expresión de
tipo int tree
```

```
S (S 3);;
- : int tree tree = S (S 3)
```

```
let t1 = S 1;;
val t1 : int tree = S 1
```

```
let t2 = S 2;;
val t2 : int tree = S 2
```

```
let t = C (3, S 5, S 2);;
val t : int tree = C (3, S 5, S 2)
```

```
S [1;2;3];;
- : int list tree = S [1; 2; 3]
```

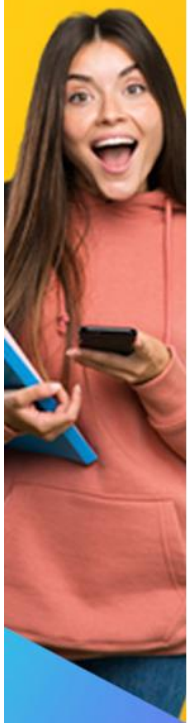
```
S [abs];;
- : (int -> int) list tree = S [<fun>]
```

```
let bigt = C (0, t2, t);;
val bigt : int tree = C (0, S 2, C (3, S 5, S 2))
```

Funcion raiz:

```
let raiz = function
        S r -> r
        |C (r, i, d) -> r;;

val raiz : 'a tree -> 'a = <fun>
```

```
raiz bigt;;
- : int = 0

raiz t;;
- : int = 3
raiz t1;;
- : int = 1
```

Función hacer ramas:

```
let ramas = function
        C (_, i, d) -> i, d;;
val ramas : 'a tree -> 'a tree * 'a tree = <fun>
ramas t;;
- : int tree * int tree = (S 5, S 2)
ramas bigt;;
- : int tree * int tree = (S 2, C (3, S 5, S 2))
let ramas = function
        C (_, i, d) -> i, d
        |_ -> raise (Failure "arbol sin ramas");;
val ramas : 'a tree -> 'a tree * 'a tree = <fun>
ramas t1;;
Exception: Failure "arbol sin ramas".
```

Funcion recursiva nodos de arbol:

```
let rec nnodos = function
        S _ -> 1
        |C (_, i, d) -> 1 + nnodos i + nnodos d;;
val nnodos : 'a tree -> int = <fun>
nnodos bigt;;
- : int = 5
```

Funcion recursiva altura de arbol:

```
let rec altura = function
        S _ -> 1
        |C (_, i, d) -> 1 + max (altura i) (altura d);;
val altura : 'a tree -> int = <fun>
altura t1;;
- : int = 1
altura t;;
- : int = 2
altura bigt;;
- : int = 3
```

Funcion de devolver nodos de un arbol como lista

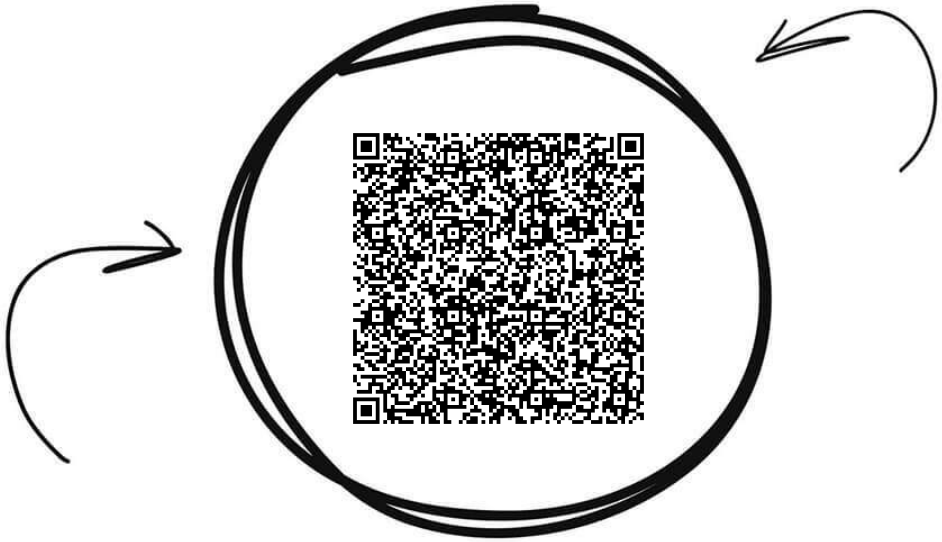
```
let rec inorden = function
```

PEKIS®
for Foodies

SUBRAYA ESTO: CÓMETE UN TACO.

RELLENOS PARA TACOS PEKIS® BRUTALES.

Paradigmas de Programación



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

```
        S r -> [r] | C (r, i, d) -> inorden i @ [r] @ inorden d;;
val inorden : 'a tree -> 'a list = <fun>

let bigt = C (9, S 2, C (0, S 1, S 21));;
val bigt : int tree = C (9, S 2, C (0, S 1, S 21))

 inorden bigt;; - : int list = [2; 9; 1; 0; 21]
- : int list = [2; 9; 1; 0; 21]
```

Función que devuelve las hojas del árbol que se le pase

```
let rec hojas = function
 S r -> [r]
 | C (_, i, d) -> hojas i @ hojas d;;
val hojas : 'a tree -> 'a list = <fun>
- : int list = [2; 1; 21]
hojas bigt;;
- : int list = [2; 9; 1; 0; 21]
```

Funcion para declarar un tipo de árbol gtree:

```
type 'a gtree =
        G of 'a * 'a gtree list;;
type 'a gtree = G of 'a * 'a gtree list

let s x = G (x, []);;
val s : 'a -> 'a gtree = <fun>

s 'D';;
- : char gtree = G ('D', [])

let r1 = G ('B', [s 'F'; s 'G']);;
val r1 : char gtree = G ('B', [G ('F', []); G ('G', [])])

let r2 = G ('C', [s 'H']);;
val r2 : char gtree = G ('C', [G ('H', [])])

let r4 = G ('E', [G ('I', [s 'J'])]);;
val r4 : char gtree = G ('E', [G ('I', [G ('J', [])])])

let t = G ('A', [r1; r2; s 'D'; r4]);;
val t : char gtree =
  G ('A',
   [G ('B', [G ('F', []); G ('G', [])]); G ('C', [G ('H', [])]); G
('D', []);
    G ('E', [G ('I', [G ('J', [])])])])
```

Función que devuelve los nodos de gtree:

```
let rec nnodos = function
        G (_, []) -> 1
```

```
        | G (_, l) -> List.fold_left (+) 1 (List.map nnodos l);;
val nnodos : 'a gtree -> int = <fun>

let rec nnodos = function
        G (_, []) -> 1
        | G (r, h::t) -> nnodos h + nnodos (G (r, t));;
val nnodos : 'a gtree -> int = <fun>

nnodos t;;
- : int = 10
```