

# Ejercicios de tipos de datos recursivos

Programación — DAW

Ricardo Pérez López

IES Doñana

Curso 2020/2021

1. La función `elem` tiene la siguiente especificación:

$$\left\{ \begin{array}{l} \text{Pre : } \text{True} \\ \text{elem}(e, t : \text{tuple}) \rightarrow \text{bool} \\ \text{Post : } \text{elem}(e, t) = \begin{cases} \text{True} & \text{si } e \text{ está en } t \\ \text{False} & \text{en caso contrario} \end{cases} \end{array} \right.$$

Escribir una función recursiva que satisfaga dicha especificación.

2. La función `cuantos` tiene la siguiente especificación:

$$\left\{ \begin{array}{l} \text{Pre : } \text{True} \\ \text{cuantos}(e, t : \text{tuple}) \rightarrow \text{int} \\ \text{Post : } \text{cuantos}(e, t) = \text{el número de veces que aparece } e \text{ en } t \end{array} \right.$$

Escribir una función recursiva que satisfaga dicha especificación y que genere un proceso:

- a) recursivo.
  - b) iterativo.
3. La función `quita` tiene la siguiente especificación:

$$\left\{ \begin{array}{l} \text{Pre : } \text{True} \\ \text{quita}(e, t : \text{tuple}) \rightarrow \text{tuple} \\ \text{Post : } \text{quita}(e, t) = \text{una tupla igual que } t \text{ pero sin los } e \end{array} \right.$$

Escribir una función recursiva que satisfaga dicha especificación y que genere un proceso:

- a) recursivo.
- b) iterativo.

4. La función `sustituye` tiene la siguiente especificación:

$$\left\{ \begin{array}{l} \text{Pre : } \text{True} \\ \text{sustituye}(a, b, t : \text{tuple}) \rightarrow \text{tuple} \\ \text{Post : } \text{sustituye}(a, b, t) = \text{una tupla igual que } t \text{ pero} \\ \text{sustituyendo los } a \text{ por } b \end{array} \right.$$

Escribir una función recursiva que satisfaga dicha especificación y que genere un proceso:

- a) recursivo.
- b) iterativo.

5. La función `ultimo` tiene la siguiente especificación:

$$\left\{ \begin{array}{l} \text{Pre : } t \neq () \\ \text{ultimo}(t : \text{tuple}) \\ \text{Post : } \text{ultimo}(t) = \text{el último elemento de } t \end{array} \right.$$

Escribir una función recursiva que satisfaga dicha especificación.

6. La función `enesimo` tiene la siguiente especificación:

$$\left\{ \begin{array}{l} \text{Pre : } t \neq () \wedge 0 \leq n < \text{len}(t) \\ \text{enesimo}(n : \text{int}, t : \text{tuple}) \\ \text{Post : } \text{enesimo}(n, t) = \text{el } n\text{-ésimo elemento de } t \end{array} \right.$$

Escribir una función recursiva que satisfaga dicha especificación.

## Soluciones

1. `elem = lambda e, t: False if t == () else \`  
    `True if t[0] == e else \`  
    `elem(e, t[1:])`

2. Definimos:

`aux = lambda a, b: 1 if a == b else 0`

a) `cuantos = lambda e, t: 0 if t == () else \`  
    `aux(e, t[0]) + cuantos(e, t[1:])`

b) `cuantos = lambda e, t: cuantos_it(e, t, 0)`  
    `cuantos_it = lambda e, t, acc: acc if t == () else \`  
    `cuantos_it(e, t[1:], acc + aux(e, t[0]))`

3. Definimos:

`aux = lambda a, b: () if a == b else (b,)`

a) `quita = lambda e, t: () if t == () else \`  
    `aux(e, t[0]) + quita(e, t[1:])`

b) `quita = lambda e, t: quita_it(e, t, ())`  
    `quita_it = lambda e, t, acc: acc if t == () else \`  
    `quita_it(e, t[1:], acc + aux(e, t[0]))`

4. Definimos:

`aux = lambda a, b, t: (b,) if a == t else (t,)`

a) `sustituye = lambda a, b, t: () if t == () else \`  
    `aux(a, b, t[0]) + sustituye(a, b, t[1:])`

b) `sustituye = lambda a, b, t: sustituye_it(a, b, t, ())`  
    `sustituye_it = lambda a, b, t, acc: \`  
    `acc if t == () else \`  
    `sustituye_it(a, b, t[1:], acc + aux(a, b, t[0]))`

5. `ultimo = lambda t: t[0] if t[1:] == () else ultimo(t[1:])`

6. `enesimo = lambda n, t: t[0] if n == 0 else enesimo(n - 1, t[1:])`