# Ejercicios de funciones recursivas

**Programación — DAW**

Ricardo Pérez López
IES Doñana

1 de noviembre de 2020

1. Dada la siguiente función matemática:

$$f(n) = \begin{cases} 0 & \text{si } n = 0 \\ 1 + 2 \cdot f(n-1) & \text{si } n > 0 \end{cases}$$

   calcular el valor de $f(3)$.

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

$$\begin{cases} \textbf{Pre}: b \geq 0 \\ \texttt{potencia(}a\texttt{: int, }b\texttt{: int) -> int} \\ \textbf{Post}: \texttt{potencia(}a\texttt{, }b\texttt{)} = a^b \end{cases}$$

   *a*) Implementar la función de forma no recursiva.

   *b*) Implementar la función de forma recursiva.

3. La función `repite` tiene la siguiente especificación:

$$\begin{cases} \textbf{Pre}: n \geq 0 \\ \texttt{repite(}s\texttt{: str, }n\texttt{: int) -> str} \\ \textbf{Post}: \texttt{repite(}s\texttt{, }n\texttt{)} = s \texttt{ * } n \end{cases}$$

   Implementar la función de forma recursiva.

4. La suma lenta es un algoritmo para sumar dos números para el que sólo necesitamos saber cuáles son el anterior y el siguiente de un número dado. El algoritmo se basa en la siguiente recurrencia:

$$suma\_lenta(a, b) = \begin{cases} b & \text{si } a = 0 \\ suma\_lenta(ant(a), sig(b)) & \text{si } a > 0 \end{cases}$$

   Suponiendo que tenemos las siguientes funciones `ant` y `sig`:

```
ant = lambda n: n - 1
sig = lambda n: n + 1
```

Se pide:

*a*) Escribir su especificación.

*b*) Implementar una función recursiva que satisfaga dicha especificación.

5. La función `suma_digitos` calcula la suma de los dígitos de un número entero:

```
suma_digitos(423) = 4 + 2 + 3 = 9
suma_digitos(7) = 0
```

Se pide:

*a*) Escribir su especificación.

*b*) Implementar una función recursiva que satisfaga dicha especificación.

*Indicación*: Recordar que `n // 10` le quita el último dígito a $n$. Además, `n % 10` devuelve el último dígito de $n$.

6. La función `voltea` le da la vuelta a un número entero:

```
voltea(423) = 324
voltea(7) = 7
```

Se pide:

*a*) Escribir su especificación.

*b*) Implementar una función recursiva que satisfaga dicha especificación.

*Indicación*: Usar la función `digitos` que devuelve la cantidad de dígitos que tiene un entero. Usar además la indicación del ejercicio anterior.

7. La función `par_positivo` determina si un número entero positivo es par:

```
par_positivo(0) = True
par_positivo(1) = False
par_positivo(27) = False
par_positivo(82) = True
```

Se pide:

*a*) Escribir su especificación.

*b*) Implementar una función recursiva que satisfaga dicha especificación.

8. La función `par` determina si un número entero (positivo o negativo) es par:

```
par(0) = True
par(1) = False
par(-27) = False
```

Se pide:

*a*) Escribir su especificación.

*b*) Implementar una función recursiva que satisfaga dicha especificación.

*c*) ¿Cómo se podría implementar una función `impar` a partir de la función `par`?

## Soluciones

1. $f(3) = 1+2\cdot f(2) = 1+2\cdot(1+f(1)) = 1+2\cdot(1+2\cdot(1+2\cdot f(0))) = 1+2\cdot(1+2\cdot(1+2\cdot 0)) = 1+2\cdot(1+2\cdot 1) = 1+2\cdot 3 = 7.$

2.

   *a*) `potencia = lambda a, b: a ** b`

   *b*) `potencia = lambda a, b: 1 if b == 0 else a * potencia(a, b - 1)`

3. `repite = lambda s, n: '' if n == 0 else s + repite(s, n - 1)`

4.

   *a*)

$$\begin{cases} \textbf{Pre}: a \geq 0 \\ \texttt{suma\_lenta(}a\texttt{: int, }b\texttt{: int) -> int} \\ \textbf{Post}: \texttt{suma\_lenta(}a\texttt{, }b\texttt{)} = a + b \end{cases}$$

   *b*) `suma_lenta = lambda a, b: b if a == 0 else suma_lenta(ant(a), sig(b))`

5.

   *a*)

$$\begin{cases} \textbf{Pre}: n \geq 0 \\ \texttt{suma\_digitos(}n\texttt{: int) -> int} \\ \textbf{Post}: \texttt{suma\_digitos(}n\texttt{)} = \text{la suma de los dígitos de } n \end{cases}$$

   *b*) `suma_digitos = lambda n: n if n < 10 else (n % 10) + suma_digitos(n // 10)`

6.

   *a*)

$$\begin{cases} \textbf{Pre}: n \geq 0 \\ \texttt{voltea(}n\texttt{: int) -> int} \\ \textbf{Post}: \texttt{voltea(}n\texttt{)} = \text{el número } n \text{ con los dígitos al revés} \end{cases}$$

   *b*)
```
voltea = lambda n: n if n < 10 else \
                   (n % 10) * 10 ** (digitos(n) - 1) + voltea(n // 10)
```

7.

   *a*)

$$\begin{cases} \textbf{Pre}: n \geq 0 \\ \texttt{par\_positivo(}n\texttt{: int) -> bool} \\ \textbf{Post}: \texttt{par\_positivo(}n\texttt{)} = \begin{cases} \texttt{True} & \text{si } n \text{ es par} \\ \texttt{False} & \text{en caso contrario} \end{cases} \end{cases}$$

   *b*)
```
par_positivo = lambda n: True if n == 0 else \
                         False if par_positivo(n - 1) else \
                         True
```

8.

*a*)

$$\begin{cases} \textbf{Pre}: \texttt{True} \\ \texttt{par}(n\texttt{: int) -> bool} \\ \textbf{Post}: \texttt{par}(n) = \begin{cases} \texttt{True} & \text{si } n \text{ es par} \\ \texttt{False} & \text{en caso contrario} \end{cases} \end{cases}$$

*b*)

```
par = lambda n: True if n == 0 else \
                False if par(abs(n) - 1) else \
                True
```

*c*)

```
impar = lambda n: not par(n)
```