Recursividad

Introducción

“Para entender la recursividad primero hay que entender la recursividad”.

En el diccionario de la RAE, la palabra “recurrencia” se define así: 1. f. Cualidad de recurrente. 2. f. Mat. Propiedad de aquellas secuencias en las que cualquier término se puede calcular conociendo los precedentes.

La recursividad está presente en muchos sistemas del mundo real. Y, por eso, muchos problemas que queremos afrontar tienen una esencia recursiva. La recursividad es una herramienta conveniente y muy útil para diseñar modelos informáticos de la realidad que deseamos modelizar. Se dice que un sistema es recursivo cuando está parcial o completamente definido en términos de sí mismo. Quizá todos nos hemos planteado alguna vez una imagen recursiva: una fotografía que muestra a una persona que sostiene entre sus manos esa fotografía, que muestra a esa persona que sostiene entre sus manos esa fotografía, que muestra a esa persona que sostiene entre sus manos esa fotografía, que muestra…

Definición

Los algoritmos **recursivos** son aquellos que se invocan, directa o indirectamente a sí mismos.

Encontramos ejemplos de invocaciones recursivas en algunas definiciones de entes matemáticos y en algoritmos. Por ejemplo, podemos definir factorial de un número natural n de la siguiente manera:

- Si n = 0, 0! = 1
- Si n > 0, n! = n * (n-1)!

Se observa que al definir factorial de un número se está basando en el factorial del número anterior. También se puede observar que esto no es así indefinidamente, sino que hay un caso particular que es el del cero que tiene su propia definición.

En C podemos escribir una función factorial que aproveche las características recursivas de la definición matemática.

Diseño de funciones recursivas

Al diseñar un algoritmo recursivo hay que identificar:

- cuál es el caso particular y cómo se resuelve (caso base).
- cómo se obtiene cada caso general a partir de uno más reducido, convergiendo hacia el caso base.

En el diseño de la función factorial identificamos:

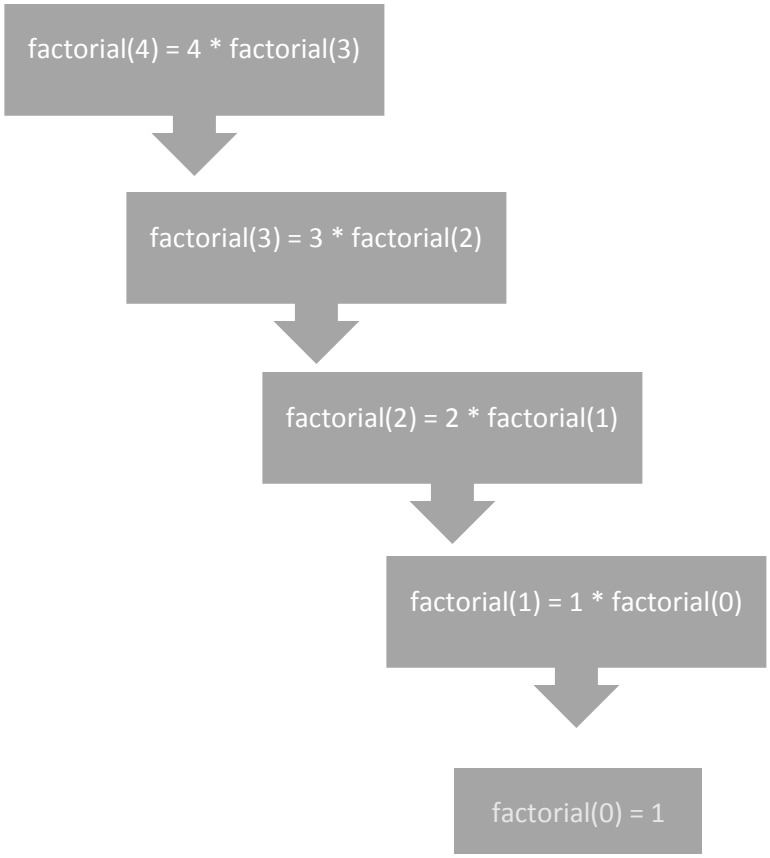
- que es una función recursiva porque el factorial de un número se puede obtener del anterior.
- hay un caso base, el 0, que siempre da 1.
- para cada número mayor que 0, el resultado se obtiene multiplicándolo por el anterior.
- siempre se finaliza en el caso base.

Por todo lo anterior, es posible diseñar la función recursiva factorial:

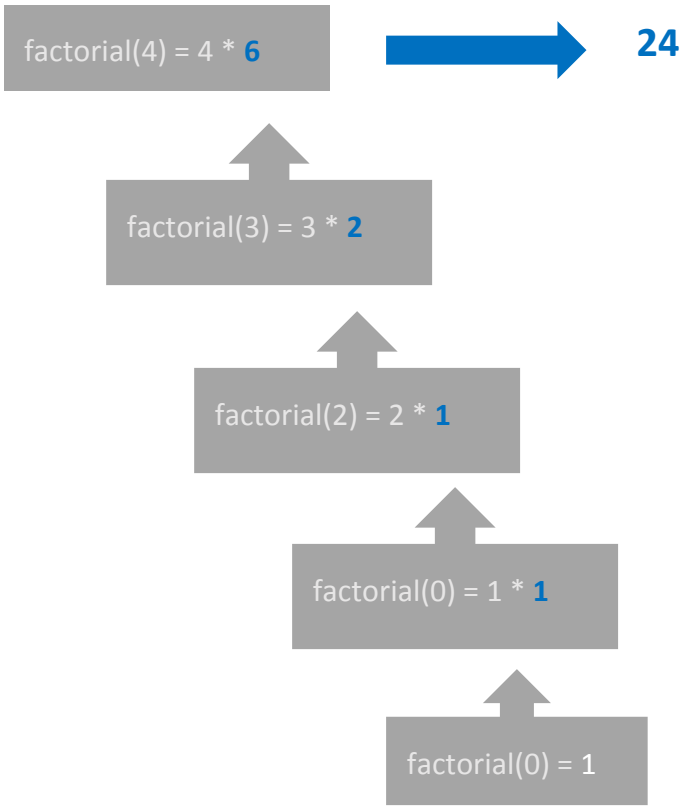
En C, dicha función resulta:

```
int factorial(int n)
{
if (n == 0)/*caso base*/
     return 1;
      else/*caso general*/
     return n * factorial (n - 1);
}
```

Si se invoca factorial con, por ejemplo 4, la secuencia de llamadas recursivas es:



Al resolver el caso base, se puede ir resolviendo las invocaciones que quedaron pendientes:



Espacio usado por un algoritmo recursivo

Si miramos la traza de las llamadas, veremos que, para que la ejecución de la función pueda recorrer el camino de vuelta, en algún lugar se deberán guardar los valores de los parámetros y de las variables locales de las llamadas anteriores. Este lugar es la pila de ejecución. Su nombre responde a que en una pila de platos solamente podemos colocar cosas encima y acceder al elemento de la cima (que es el último que hemos introducido). Debido a que en la ejecución de un método recursivo podemos tener que realizar muchas llamadas hasta alcanzar los casos simples, puede darse la situación de que se llena la pila de ejecución y se produzca un error de desbordamiento de pila (Stack Overflow).