# Ejemplo de recursividad, cálculo del factorial.

Cuando se explicó como se definen funciones en C++, no se estableció ninguna restricción a las instrucciones que pueden escribirse en el cuerpo de una función. Por lo tanto es posible escribir en el cuerpo de una función llamadas a la misma función. Una llamada a una función desde la misma función se denomina **llamada recursiva** y se puede decir que la función es recursiva o bien que utiliza la recursividad.
La recursividad es la forma más sencilla de escribir en un lenguaje de programación la definición de acciones inductivas. Un ejemplo de una acción inductiva es el cálculo del factorial de un número natural:

$$x! = \begin{cases} 1 \text{ si x=0} \\ \text{x(x-1)! si x>0} \end{cases}$$

En C++ se podría escribir esa acción de la siguiente forma:

```
int factorial(int x)
{
if (x==0) return 1;
else return x*factorial(x-1);
}
```

En la figura 1 se muestra un diagrama con las sucesivas llamadas recursivas a la función factorial y el retorno de las mismas. Los pasos que sigue el programa se numeran desde 0 a 7. A continuación se explica cada uno de ellos:

- Paso 0: el valor del parámetro real se copia en el parámetro formal x. Como su valor no es cero, se ejecuta la sentencia incluida en el `else`. Dicha sentencia incluye una llamada recursiva a `factorial` con el valor 2.
- Paso 1: el valor 2 se copia en x, del mismo modo que en el caso anterior, se ejecuta la sentencia incluida en el `else`, produciéndose otra llamada recursiva a `factorial`, en este caso con el valor 1.
- Paso 2: el valor 1 se copia en x, como ocurría en el caso anterior, se ejecuta la sentencia incluida en el `else`, produciéndose otra llamada recursiva a `factorial`, en este caso con el valor 0.
- Paso 3: el valor 0 se copia en x. En este caso se cumple la condición del `if`, de modo que se retorna 1 y no se efectúa ninguna llamada recursiva.
- Paso 4: el valor de retorno 1 sustituye a la llamada `factorial(1-1)`, de modo que se puede calcular la expresión `1*factorial(1-1)`, su valor es 1.

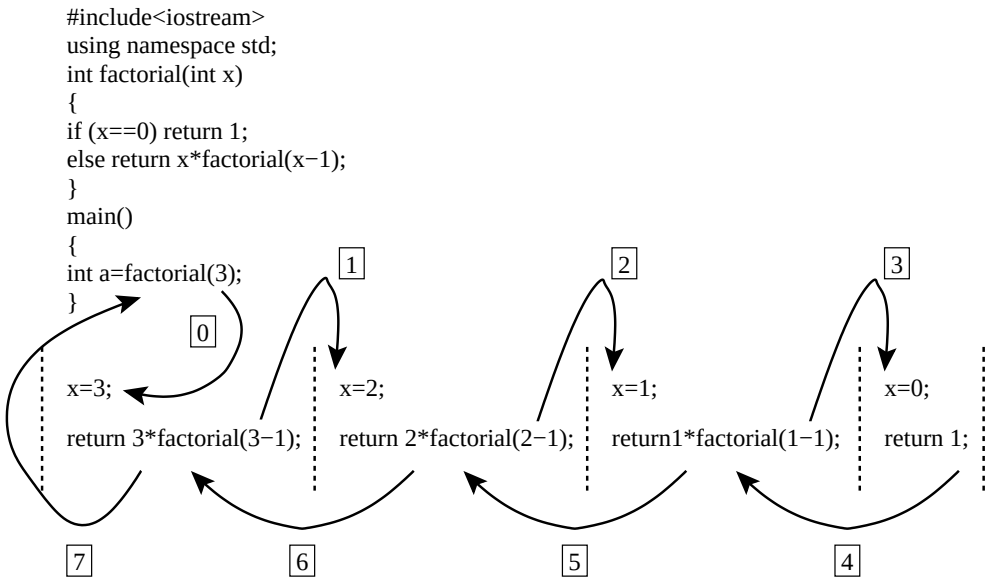


Figura 1: Diagrama de las llamadas recursivas a la función factorial.

- Paso 5: el valor de retorno 1 sustituye a la llamada `factorial(2-1)`, de modo que se puede calcular la expresión `2*factorial(2-1)`, su valor es 2.
- Paso 6: el valor de retorno 2 sustituye a la llamada `factorial(3-1)`, de modo que se puede calcular la expresión `3*factorial(3-1)`, su valor es 6.
- Paso 7: el valor de retorno 6 sustituye a la llamada `factorial(3)` y se asigna a la variable `a`, el programa termina.

La función anterior podría haberse escrito sin utilizar la recursividad, en forma iterativa, del siguiente modo:

```
int factorial(int x)
{
int f=1;
for (int i=2;i<x;i++)
  f=f*x;
return f;
}
```

Las dos funciones realizan el mismo cálculo utilizando las mismas operaciones, pero la forma recursiva emplea mayor cantidad de memoria (el paso del parámetro es por valor) y además existe el coste añadido de las sucesivas llamadas, las cuales conllevan un determinado tiempo de procesador. En general una función no es recursiva o iterativa intrínsecamente: para cada función recursiva existe una función iterativa con su misma especificación, y viceversa. Es frecuente codificar de forma recursiva una función cuando su versión iterativa es muy compleja.