Grado en Estadística Aplicada

Programación II

Recursividad

José Javier Galán Hernández
josejgal@ucm.es

FACULTAD ESTUDIOS ESTADISTICOS

DEFINICION

Función que se llama a si misma durante su propia ejecución.

Se debe planificar el momento en que dejan de llamarse a sí mismas o tendremos una función recursiva infinita.

Es común usarlas para dividir una tarea en sub-tareas más simples de forma que sea más fácil abordar el problema y solucionarlo.

Necesitamos conocer la solución no recursiva para algún caso sencillo (denominado caso base) y hacer que la división de nuestro problema acabe recurriendo a los casos base que hayamos definido.

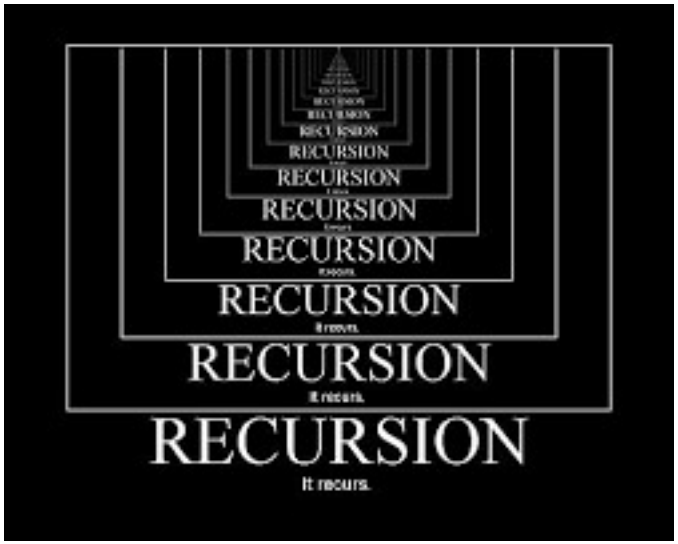
FACULTAD ESTUDIOS ESTADISTICOS

DISEÑO

Se descompone el problema en problemas mas pequeños

Resolvemos el problemas para tener un caso base (uno por lo menos)

Construimos la solución final a partir de las soluciones parciales.



PASOS

1. Resolver los casos base:
   - Sin recursividad.
   - Debe existir algún caso base.

2. Solución para el caso general:
   - Expresión de forma recursiva.
   - Puede contener instrucciones complementarias.

FACULTAD ESTUDIOS ESTADISTICOS

Iterativo vs Recursivo

- Consumo de CPU y espacio en memoria asociados a las llamadas recursivas.
- La redundancia (algunas soluciones recursivas resuelven un problema en repetidas ocasiones).
- La complejidad de la solución (en ocasiones, la solución iterativa es muy difícil de encontrar).
- La sencillez, legibilidad y limpieza del código resultante de la solución recursiva del problema.

EJEMPLO

FACTORIAL

Obtener el valor factorial de un numero quiere decir que hay que multiplicar todos los números enteros positivos que hay entre ese numero y el 1.

La función factorial se representa con el símbolo exclamación “!”.

4! = 1 x 2 x 3 x 4 = 24

El factorial de 1 y 0 es 1.

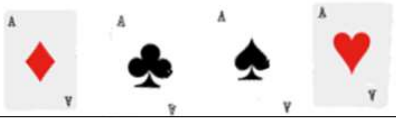
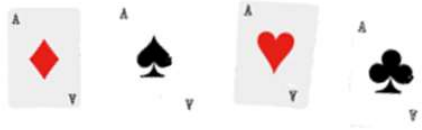
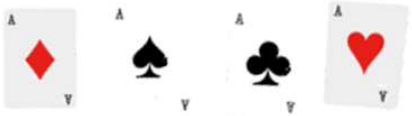
1! = 1 0! = 1 (por convenio ya que 0 * 1 = 0)

FACULTAD ESTUDIOS ESTADISTICOS

EJEMPLO

FACTORIAL

Pepa ha sacado los 4 ases de una baraja.
Va a colocarlos en fila encima de la mesa.
¿De cuántas maneras distintas podría colocarlos?

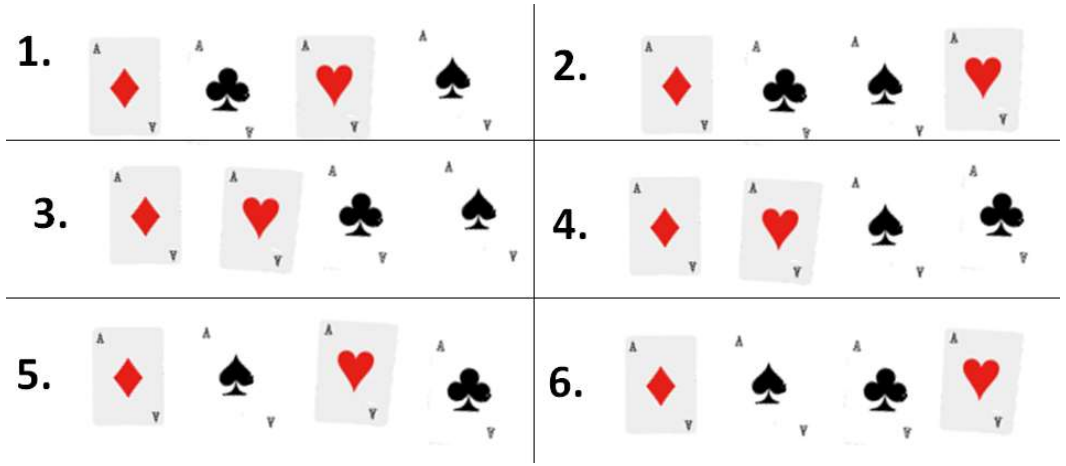
1. 	2. 
3. 	4. 
5. 	6. 

FACULTAD ESTUDIOS ESTADISTICOS

EJEMPLO

FACTORIAL

En este problema nos están pidiendo lo que se llama una **permutación**, es decir, que averigüemos todas las maneras posibles en las que estas 4 cartas se pueden combinar **teniendo en cuenta el orden** en el que las colocamos.
Si comenzamos haciendo todas las filas posibles comenzando con el as de diamantes, podemos hacer 6 combinaciones:



EJEMPLO

FACTORIAL

También tendremos 6 combinaciones posibles con el de tréboles, con el de corazones y con el de picas, es decir, 6 combinaciones empezando con cada una de las 4 cartas: 4 x 6 = 24

SOLUCIÓN: Podría colocarlos de 24 maneras posibles

EJEMPLO

FACTORIAL

Utilizando la función factorial, podríamos haber resuelto el problema de forma mucho más sencilla:

Pensamos en una sola combinación de los **4 ases**:
– Cuando hemos elegido el primero, ya solo **nos quedan 3** para elegir
– Cuando hemos elegido el segundo, ya solo **nos quedan 2** para elegir
– Cuando hemos elegido el tercero, ya solo **nos queda 1** para elegir
Por lo tanto, todas las combinaciones posibles serán 4 x 3 x 2 x 1.
O lo que es lo mismo, 4! = 24

EJEMPLO

FACTORIAL

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

Forma iterativa.

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

Forma recursiva.

FACULTAD ESTUDIOS ESTADISTICOS

EJEMPLO

FACTORIAL

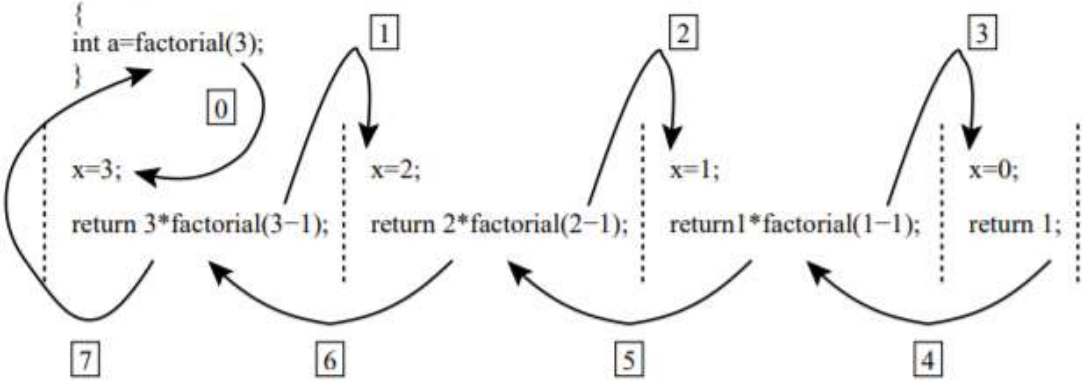
$$x! = \begin{cases} 1 \text{ si } x=0 \\ x(x-1)! \text{ si } x>0 \end{cases}$$

Forma iterativa.

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

Forma recursiva.

```
#include<iostream>
using namespace std;
int factorial(int x)
{
if (x==0) return 1;
else return x*factorial(x-1);
}
main()
{
int a=factorial(3);
}
```



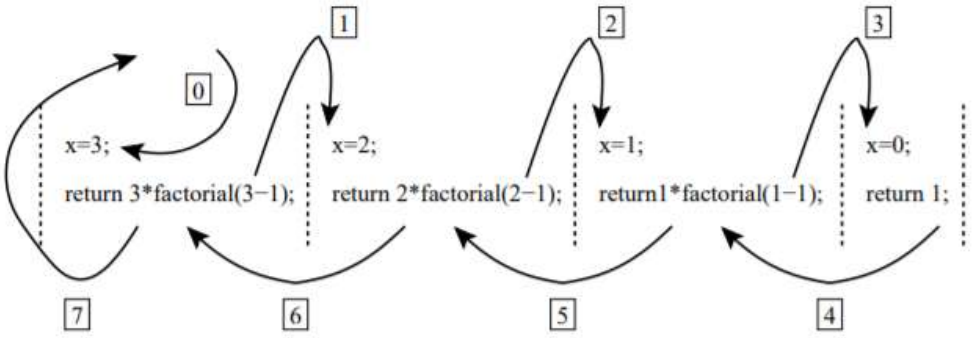
FACULTAD ESTUDIOS ESTADISTICOS

EJEMPLO

FACTORIAL Forma recursiva.

Paso 0. X obtiene el valor original 3 y como no es 0 devuelve el valor de la función.
Paso 1. X obtiene el valor 2 y como no es 0 devuelve el valor de la función.
Paso 2. X obtiene el valor 1 y como no es 0 devuelve el valor de la función.
Paso 3. X obtiene el valor 0 y como si es 0 devuelve 1.
Paso 4. El valor devuelto es 1, calcula 1*factorial(1-1) que es 1.
Paso 5. El valor devuelto es 1, calcula 2*factorial(2-1) que es 2.
Paso 6. El valor devuelto es 2, calcula 3*factorial(3-1) que es 6.
Paso 7. El valor devuelto es 6, se asigna a la variable y la devuelve.

```
#include<iostream>
using namespace std;
int factorial(int x)
{
if (x==0) return 1;
else return x*factorial(x−1);
}
main()
{
int a=factorial(3);
```



FACULTAD ESTUDIOS ESTADISTICOS

EJERCICIOS

1. Implementa una función recursiva que calcule la suma de los n primeros números naturales. La cabecera sería la siguiente:
   unsigned sumanaturales(unsigned n) donde el parámetro n es el número de naturales a sumar.

2. El valor de la función potencia x^n, se puede definir recursivamente del modo siguiente:
   x^n =1 si n=0
   x^n =x*x^(n-1) si n>=1
   Implementa una función potencia que calcule recursivamente el valor de xn con la siguiente cabecera:
   unsigned potencia(unsigned x, unsigned n)

3. Implementa una función recursiva que calcule el producto de dos número naturales x e y. La cabecera sería la siguiente:
   unsigned producto(unsigned x, unsigned y)
   A la hora de diseñar la solución ten en cuenta que los únicos operadores aritméticos que puedes usar son la suma y la resta.

4. Implementa un procedimiento recursivo que imprima los dígitos de un número natural n en orden inverso. Por ejemplo, para n=675 la salida debería ser 576. La cabecera sería la siguiente: void inverso (unsigned n)

5. Implementa una función recursiva que devuelva true si el número que se le pasa como parámetro es primo y false en caso contrario. La cabecera de la función sería la siguiente:
   bool esPrimo (unsigned num, unsigned divisor)
   en el primer parámetro le habríamos de pasar el número, y el segundo parámetro es un número para que hay que comprobar si es o no divisor. Inicialmente (en la llamada a la función desde main) el valor de ese parámetro es 2.

6. Implementa un procedimiento recursivo al que se le pase como parámetro un número n en base 10 (decimal), e imprima su valor en base 2 (binario). La cabecera sería la siguiente:
   void decimalAbinario (unsigned n)
   Por ejemplo para n=23, el procedimiento debería imprimir 10111.