La Salle Gràcia

Programación

Módulo 03

UF2- Recursividad

M03 – PROGRAMACIÓN

Contenidos

- **UF1 – Programación Estructurada**
- UF2 – Diseño Modular
- 2.3 Recursividad
- UF3 – Fundamentos de Gestión de Archivos

Recursividad

- Una función recursiva es aquella que se llama a sí misma.
- La recursividad es una alternativa a la repetición o iteración.
  - En tiempo de computadora y ocupación de memoria, la solución recursiva es menos eficiente que la iterativa, existen situaciones en las que la recursividad es una solución simple y natural a un problema que en otro caso será difícil de resolver

Recursividad

La característica principal de la recursividad es que siempre existe un medio de salir de la definición (caso base o salida), y la segunda condición (llamado recursivo) es propiamente donde se llama a sí misma.

Una función recursiva simple en Java es:

```
public void infinito ()
{
        infinito ();
}
```

Definición

De manera más formal, una función recursiva es invocada para solucionar un problema y dicha función sabe cómo resolver los casos más sencillo (**casos bases**).

Donde si la función es llamada desde un caso base, ésta simplemente devuelve el resultado.

Si es llamada mediante un problema más complejo, la función lo divide en dos partes conceptuales: una parte de dicha función sabe resolver y otra que no sabe resolver.

Ejemplo

¿Cuántas formas hay de colocar n objetos en orden?

- Podemos colocar cualquiera de los n objetos en la primera posición.
- A continuación colocamos los (n -1) objetos restantes.
- Por tanto: P(n)=n*P(n-1)=n!

Ejemplo

El factorial de un entero no negativo n, esta definido como:

$$n! = n * (n\text{-}1) * (n\text{-}2) * \ldots * 2 * 1$$

Donde 1! es igual a 1 y 0! se define como 1.
El factorial de un entero k puede calcularse de manera iterativa como sigue:

```
fact  = 1;

for (int i = k; i>=1; i--)
    fact *= i;
```

Ejemplo

Ahora de manera recursiva se puede definir el factorial como:

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

Donde el caso base es: 1 Si n = 0.

El caso general es: n * (n-1)! Si n > 0.

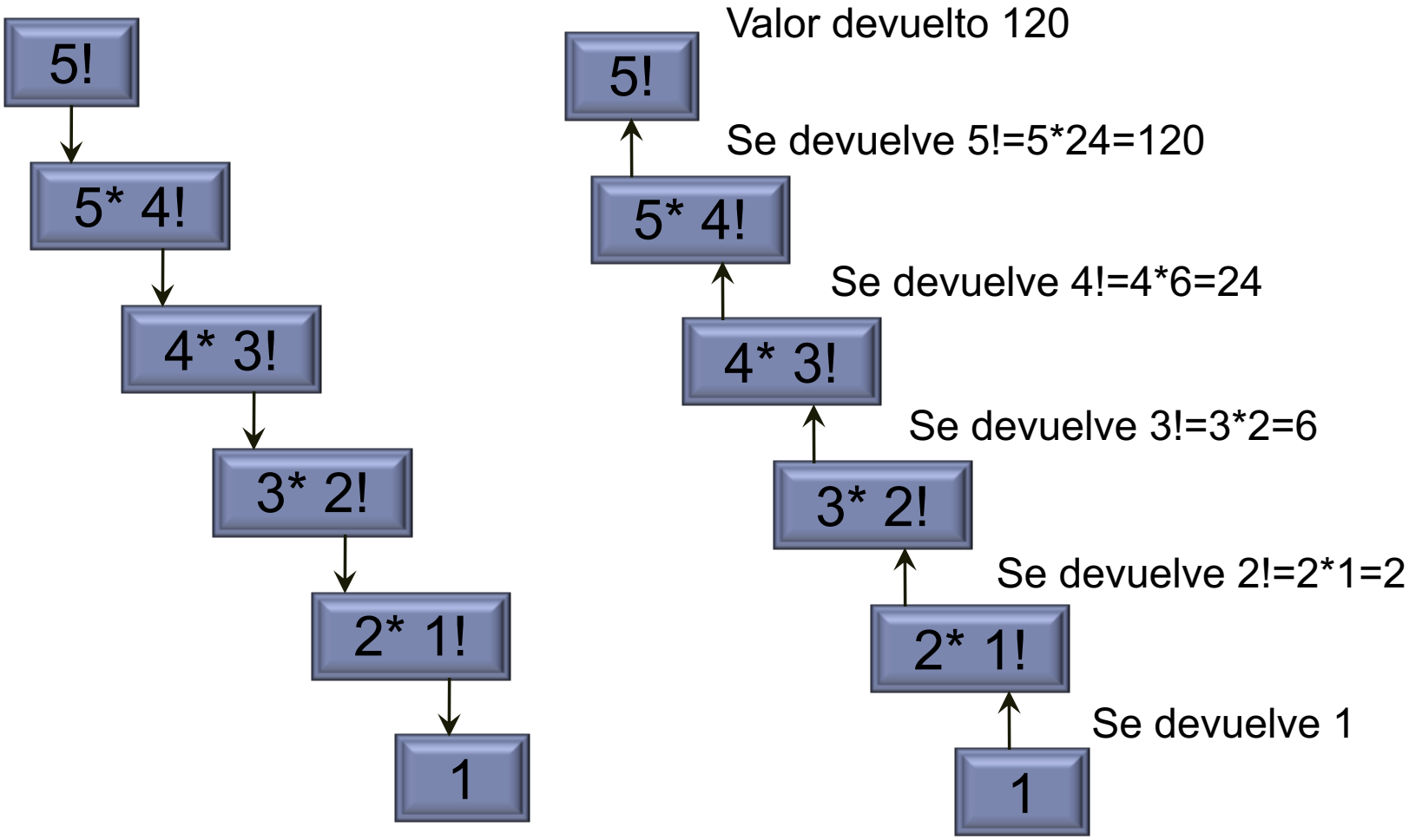
Por ejemplo si se quiere calcular el factorial de 5, se tendría:

5! = 5*4*3*2*1

5! = 5*(4*3*2*1)

5! = 5*4!

Ejemplo

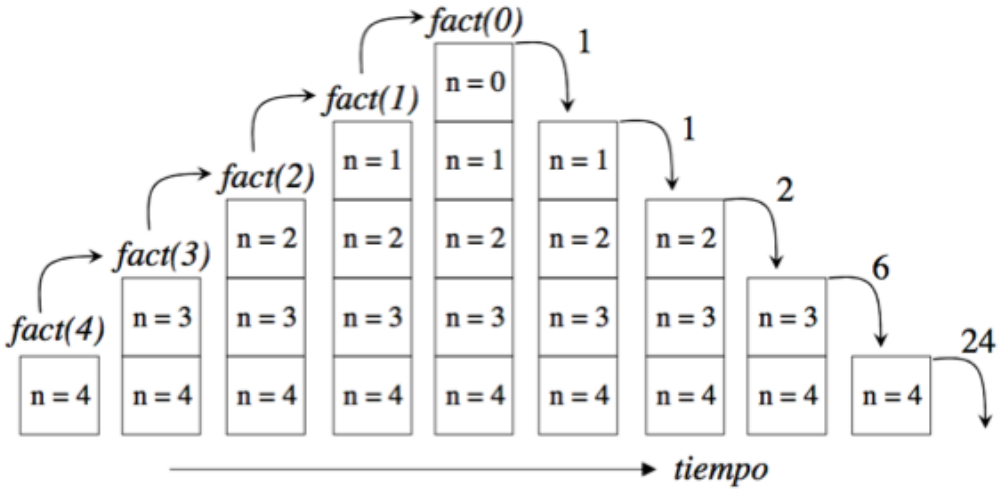

5!
5* 4!
4* 3!
3* 2!
2* 1!
1
Valor devuelto 120
5!
Se devuelve 5!=5*24=120
5* 4!
Se devuelve 4!=4*6=24
4* 3!
Se devuelve 3!=3*2=6
3* 2!
Se devuelve 2!=2*1=2
2* 1!
Se devuelve 1
1

Solución en JAVA

```
static int factorial (int n)
{
  int resultado;

  if (n==0)                              // Caso base
     resultado = 1;
  else                                   // Caso general
     resultado = n*factorial(n-1);

  return resultado;
}
```

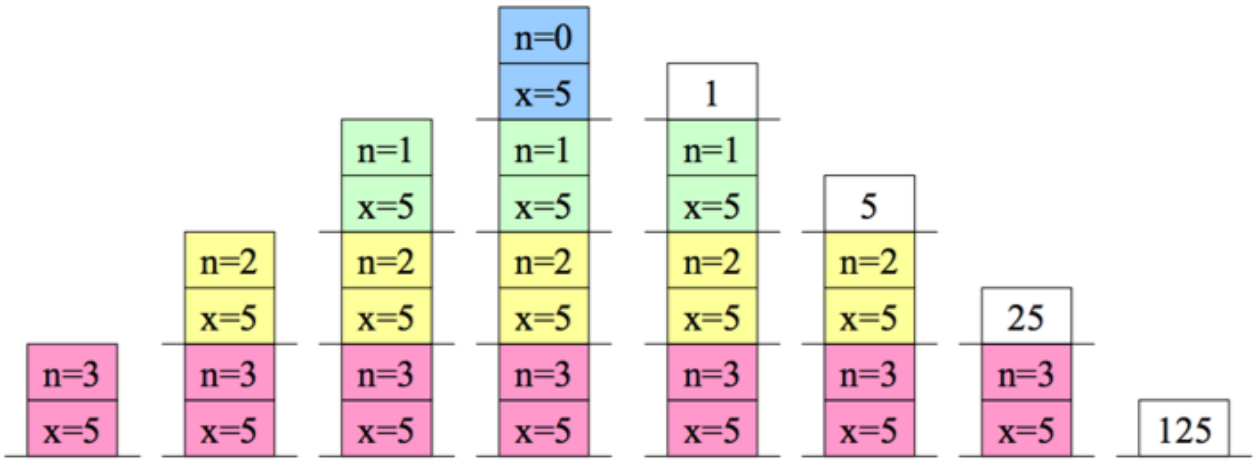


Otro Ejemplo

Calcular la potencia de un numero x de manera recursiva.

Otro Ejemplo

```
static int potencia (int x, int n)
{
  if (n==0)                          // Caso base
     return 1;
  else                               // Caso general
     return x * potencia(x,n-1);
}
```



Cálculo de 5^3

Diseño de Algoritmos Recursivos

1. Resolución e problema para los casos bases.
   - Sin emplear recursividad.
   - Siempre debe existir un caso base.

2. Solución para el caso general.
   - Expresión de forma recursiva.
   - Pueden incluirse pasos adicionales (para combinar las soluciones parciales).