



Universidad de Granada

RECURSIVIDAD



DECSAI
Departamento de Ciencias
de la Computación e I.A.
Universidad de Granada

Introducción a la Recursividad

2

- La recursividad es una herramienta muy útil en matemáticas y programación. Permite definir conceptos y algoritmos de una forma simple. Por ejemplo:

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

- Una función es recursiva si, dentro de su código fuente, se llama a sí misma (en **términos más pequeños**).

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

- Toda función recursiva tiene 2 componentes:
 - **Caso base:** Contiene la condición de parada de la recursividad.
 - **Caso general:** Contiene el código recursiv

Siempre debe existir al menos un caso base




Cálculo del Factorial de 3

3


$$3! = 3 * 2!$$


(1)

$$3! = 3 * \boxed{2!}$$


$$2! = 2 * 1!$$


(2)


$$3! = 3 * \boxed{2!}$$



$$2! = 2 * \boxed{1!}$$


$$1! = 1 * 0!$$

(3)

$$3! = 3 * \boxed{2!}$$


$$2! = 2 * \boxed{1!}$$


$$1! = 1 * \boxed{0!}$$


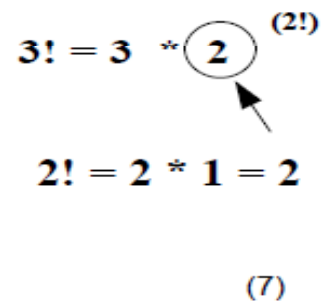
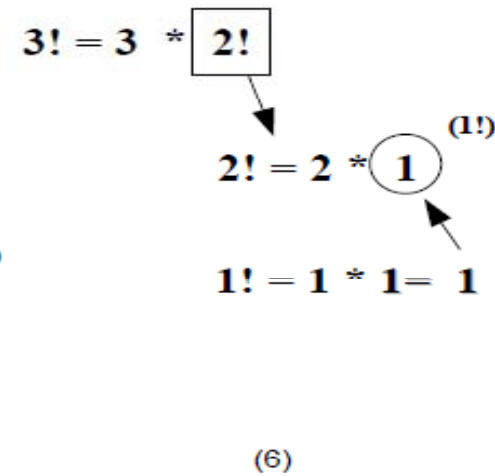
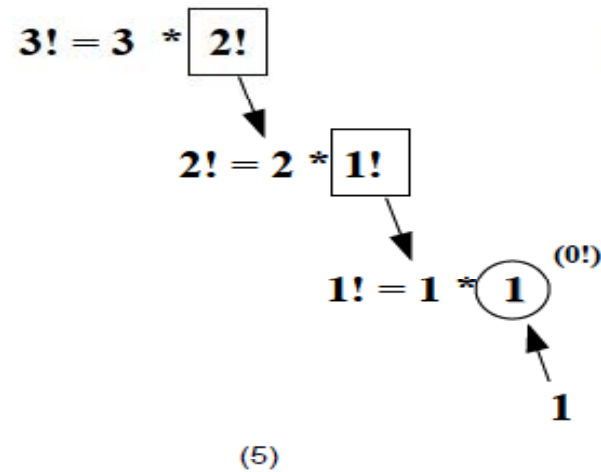
$$0! = 1$$

(CASO BASE)

(4)

Cálculo del Factorial de 3

4



$3! = 3 * 2 = 6$

(8)

Ejemplos de Funciones Recursivas

5

Ejemplo de cálculo recursivo de varias funciones conocidas:

$$Suma(a, b) = \begin{cases} a; & \text{si } b = 0 \\ 1 + Suma(a, b - 1); & \text{si } b > 0 \end{cases}$$

$$Multiplica(a, b) = \begin{cases} 1; & \text{si } b = 0 \\ a + Multiplica(a, b - 1); & \text{si } b > 0 \end{cases}$$

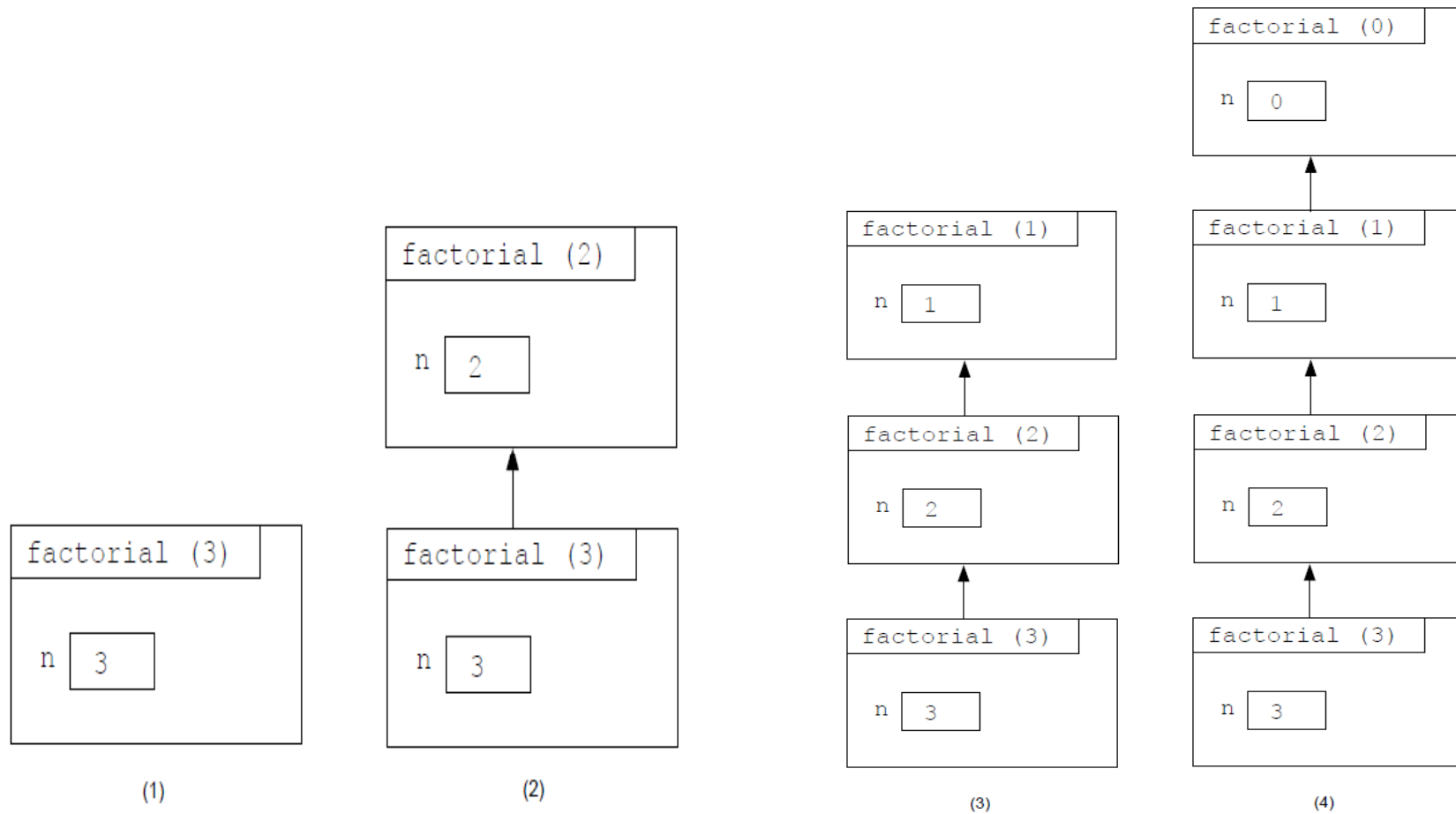
$$Maximo(V, n) = \begin{cases} v[0]; & \text{si } n = 0 \\ \max(V[n] \text{ ó } \maximo(V, n - 1)); & \text{si } n > 0 \end{cases}$$

$$ElevaA(x, n) = \begin{cases} 1; & \text{si } n = 0 \\ x * ElevaA(x, n - 1) & \text{si } n > 0 \end{cases}$$

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

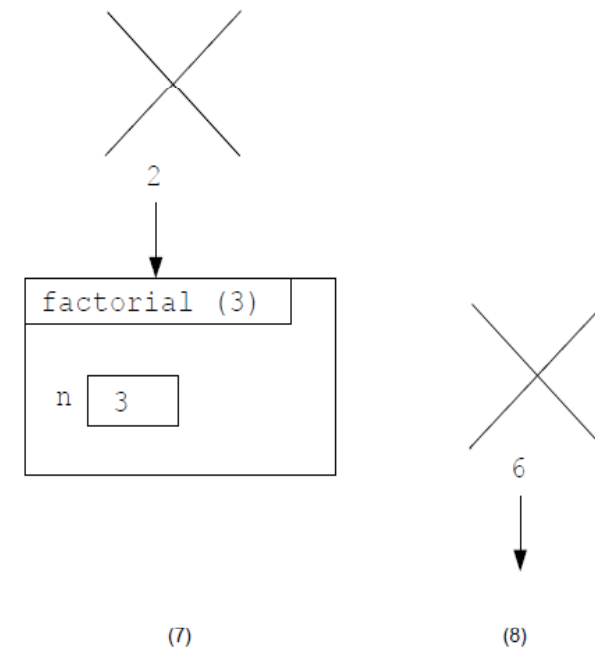
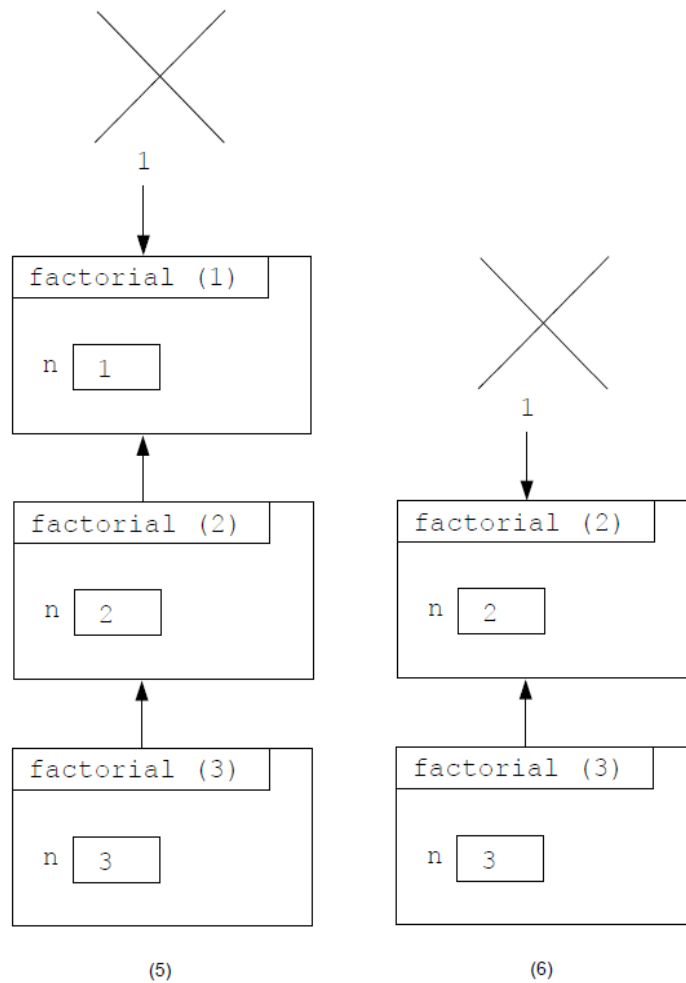
Ejecución de Funciones Recursivas

6



Ejecución de Funciones Recursivas

7



Cada llamada a una función recursiva genera un marco de trabajo en la pila. Cada marco tiene su propia copia del parámetro formal.

Búsqueda Binaria Recursiva

8

Definición recursiva como **método** de la clase **MiVector**

```
int BusquedaBinaria(int izda, int dcha, int buscado)
{
    int centro;
    if (izda > dcha)
        return -1;
    else
    {
        centro = (izda + dcha) / 2;
        if (vector_privado[centro] == buscado)
            return centro;
        else
            if (vector_privado[centro] > buscado)
                return BusquedaBinaria(izda, centro-1, buscado);
            else
                return BusquedaBinaria(centro + 1, dcha, buscado);
    }
}
```


Invertir un Vector (Entre las posiciones izda y dcha)

9

```
MiVector Invertir(int izda, int dcha)
{
    MiVector inverso;

    if (izda < dcha)
    {
        inverso = Invertir(izda+1, dcha);
        inverso.Aniade(vector_privado[izda]);
    }
    else
        if (izda == dcha)
            inverso.Aniade(vector_privado[izda]);

    return inverso; //Si izda>dcha, devuelve vector vacío.
}
```

Ven/Des de la Recursividad

10

- Ventajas de la recursividad:
 - ▣ La solución recursiva suele ser **concisa, legible y elegante** (recorrido de estructuras complejas como árboles, listas, grafos, etc).
- Desventajas de la recursividad:
 - ▣ **Alta carga computacional** (tiempo-espacio) asociada a la llamada a una función y al retorno a la función que hace la llamada.
 - ▣ Algunas soluciones recursivas pueden hacer que la solución para un determinado tamaño del problema **se calcule varias veces** (ej: Fibonacci).