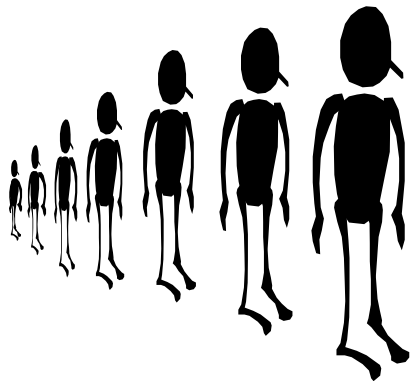
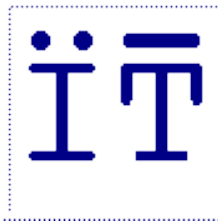


# Recursión

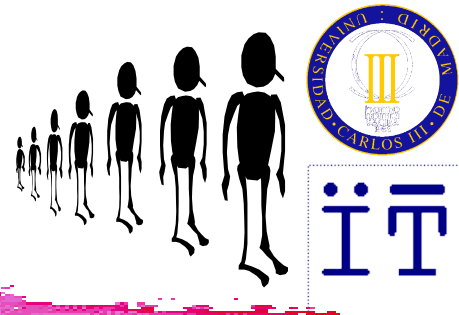


**Carlos Delgado Kloos**

**Ingeniería Telemática**

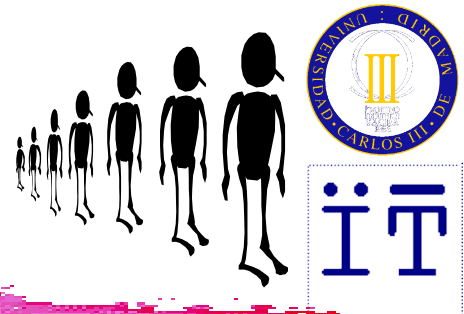
**Univ. Carlos III de Madrid**





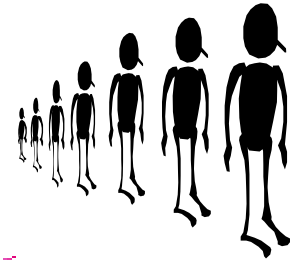
# Método recursivo

- Un método recursivo es aquel que (directa o indirectamente) se llama a si mismo.
- (Para que el método recursivo defina una computación que termina) la(s) llamada(s) recursiva(s) han de ser más sencilla(s) (de acuerdo con alguna métrica)



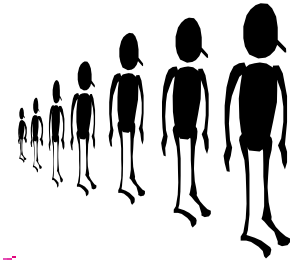
# Ejemplo 1

```
public static long s (int n)
{if (n==1)
    return 1;
else
    return s(n-1)+n;
}
```



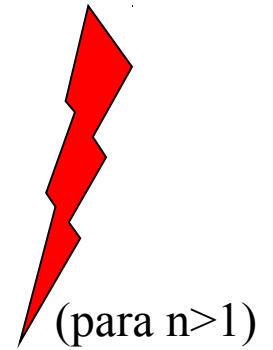
# Ejemplo 1

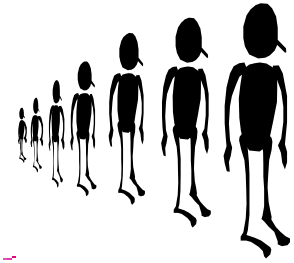
- $s(3) =$  (llamada recursiva)
- $s(2) + 3 =$  (llamada recursiva)
- $(s(1) + 2) + 3 =$  (llamada recursiva)
- $(1 + 2) + 3 =$  (suma)
- $(3 + 3) =$  (suma)
- 6



# Ejemplo 2

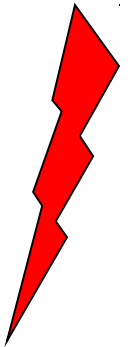
```
public static long s (int n)
{if (n==1)
    return 1;
else
    return s(n+1)+n;
}
```



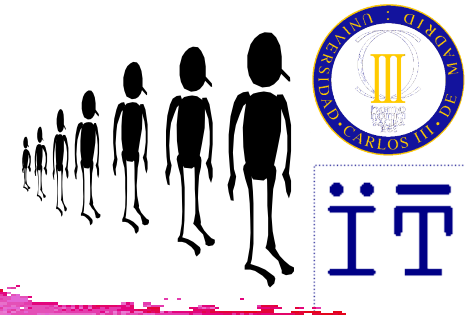


# Ejemplo 2

- $s(3) =$  (llam. rec.)
- $s(4) + 3 =$  (llam. rec.)
- $(s(5) + 4) + 3 =$  (llam. rec.)
- $((s(6) + 5) + 4) + 3 =$  (llam. rec.)
- $(( (s(7) + 6) + 5) + 4) + 3 =$  (llam. rec.)
- $(( ( (s(8) + 7) + 6) + 5) + 4) + 3 =$
- ...
- No termina



# ¿Qué debe tener un método recursivo?



## ■ Condicional

❖ Caso base (no recursivo) 1

❖ Caso recursivo  $s(n-1) + n$

(que se aproxima hacia  
la condición del caso base)  $(n==1)$

# Ejercicios: cuentaAtras

```
void cuentaAtras (int contador)
{if(contador == 0)
    return;
else {
    System.out.println(""+contador);
    cuentaAtras(--contador);
    return;
}
}
```



# Ejercicio: cuadrado

- $(N-1)^2 = N^2 - 2N + 1$
- $N^2 = (N-1)^2 + 2N - 1$
  
- $\text{cuadrado}(1) = 1$
- $\text{cuadrado}(N) = \text{cuadrado}(N-1) + 2N - 1$

# Ejercicio: cuadrado

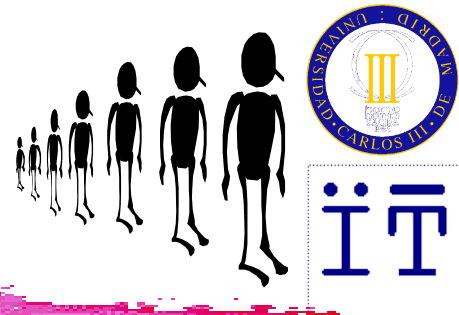
```
int cuadrado (int n)
{if (n == 1)
    return 1;
else
    return cuadrado (n-1) + 2*n-1;
}
```

# Ejercicio: misterio

- $\text{misterio}(0, Q) = Q$
- $\text{misterio}(P, Q) = \text{misterio}(P-1, Q+1)$
- ¿Cuánto vale  $\text{misterio}(2, 4)$ ?

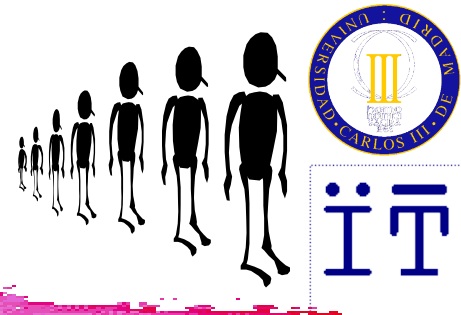
# Tipos de recursión:

## Recursión lineal



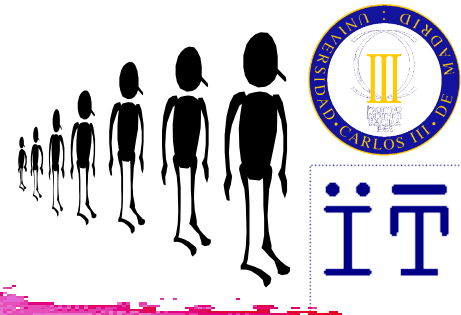
- Recursión lineal  
(máximo una llamada recursiva por rama del condicional)
  - ❖ Recursión por la cola  
(última operación en rama: llamada recursiva)
  - ❖ Recursión no por la cola  
(operación pendiente)

# Tipos de recursión: Recursión no lineal



- Recursión no lineal
  - ❖ Recursión en cascada  
(  $op(f..., f...)$  )
  - ❖ Recursión anidada  
(  $f(...f...)$  )
  - ❖ ...

# Ejemplo 3: Factorial



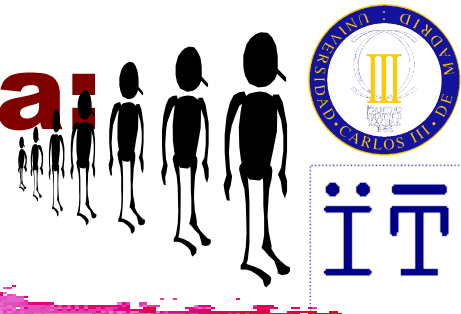
$$\text{fac}(n) = n!$$

$$\text{fac}(5) = 5 * 4 * 3 * 2 * 1$$

$$\begin{aligned} \text{fac}(5) &= \\ 5 * \text{fac}(4) &= \\ 5 * 24 &= 120 \end{aligned}$$

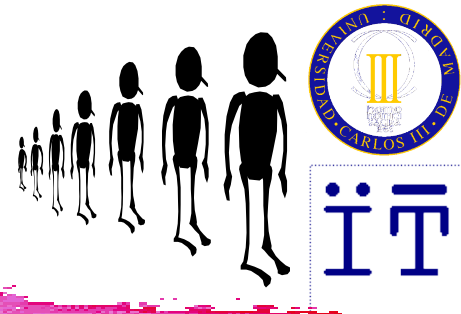
n	fac(n)
0	1
1	1
2	2
3	6
4	24
5	120
...	...

# Recursión no por la cola: Factorial



```
public static long fac (int n)
{if (n<=1)
    return 1;
else
    return n*fac(n-1);
}
```

# Recursión por la cola: Factorial

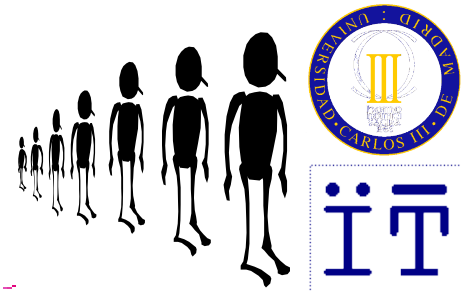


- ```
public static long fact (int n,m)
{if (n<=1)
    return m;
else
    return fact(n-1,n*m) ;
}

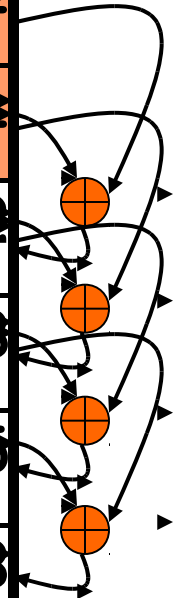
public static long fac (int n)
{return fact(n,1);}
```



# Ejemplo 4: Fibonacci



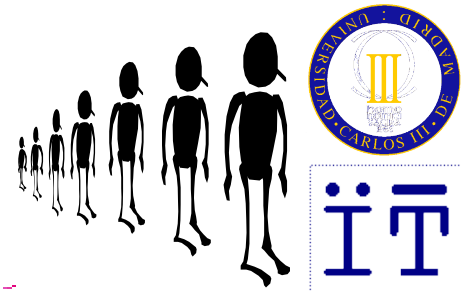
| n   | fib (n) |
|-----|---------|
| 0   | 1       |
| 1   | 1       |
| 2   | 2       |
| 3   | 3       |
| 4   | 5       |
| 5   | 8       |
| ... | ...     |



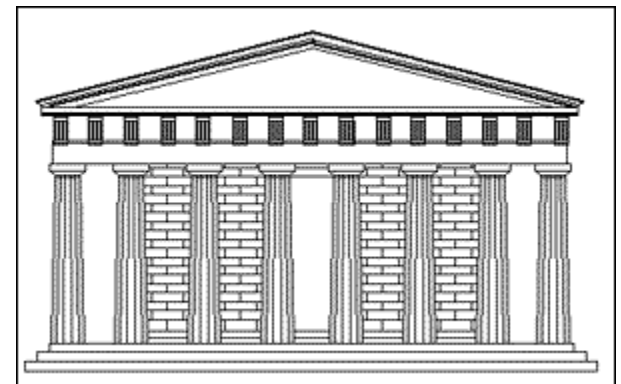
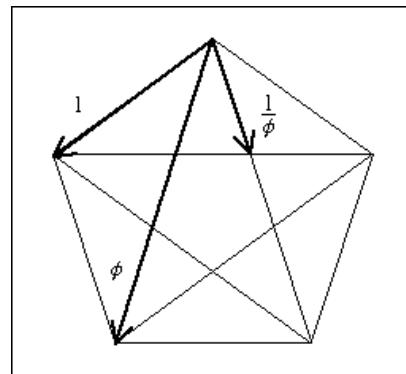
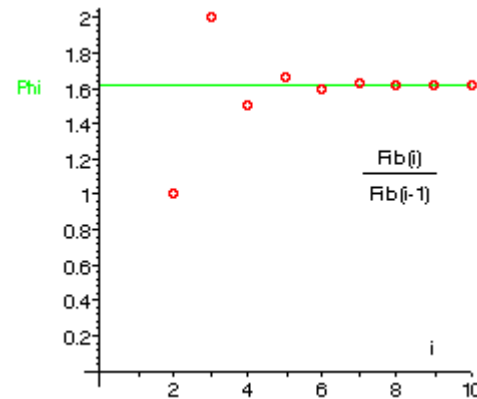
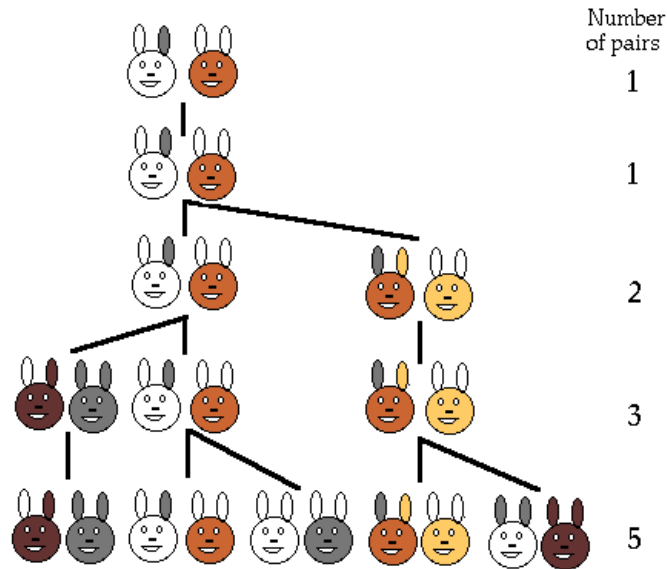
$$\begin{aligned} \text{fib}(5) &= \\ \text{fib}(4) + \text{fib}(3) &= \\ &= 5 + 3 \end{aligned}$$



# Ejemplo 4: Fibonacci

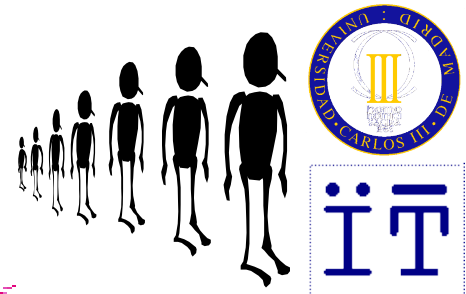


**Ejercicio:** Buscar las aplicaciones de *Fibonacci* buscando con *google*



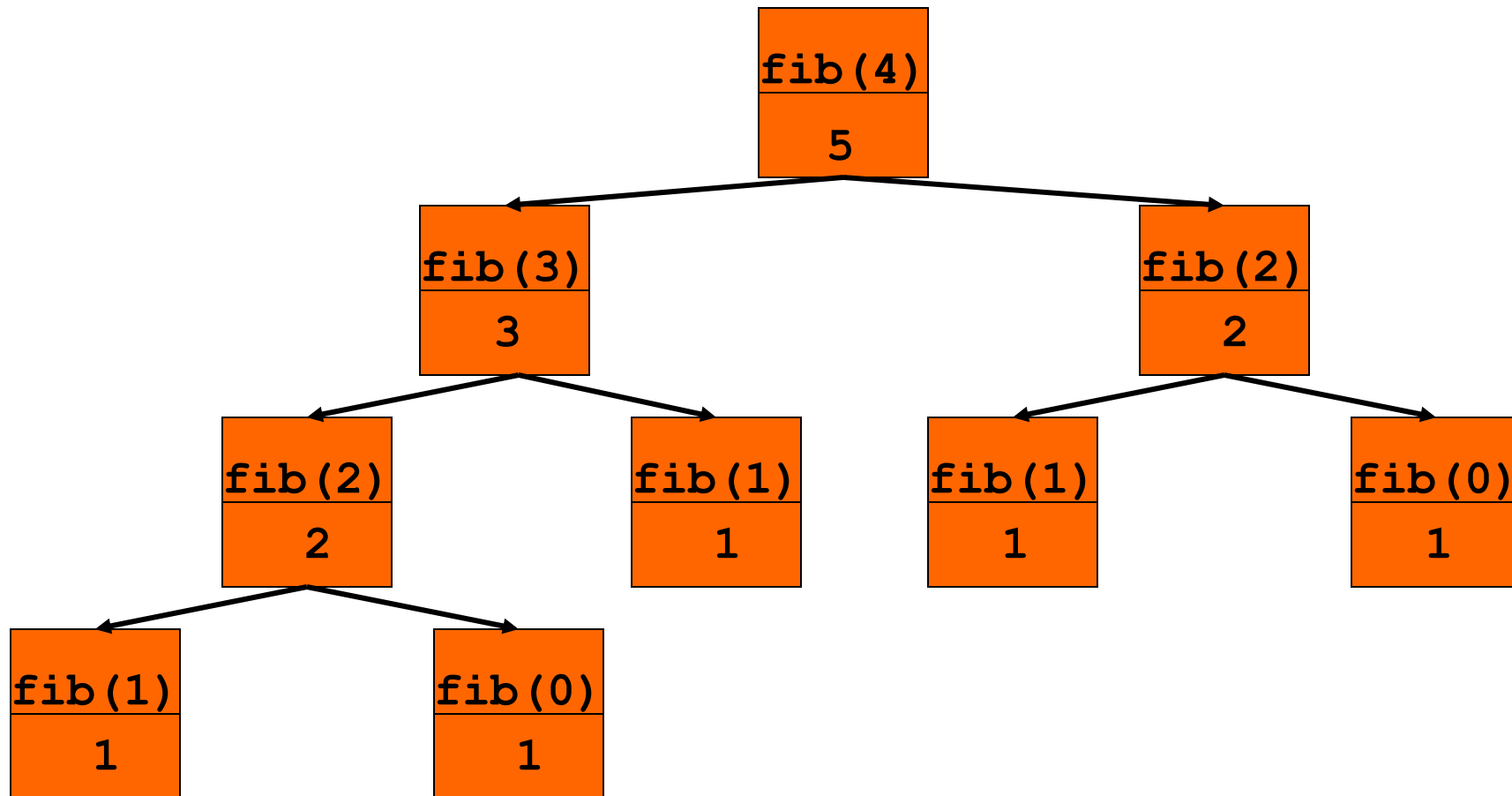
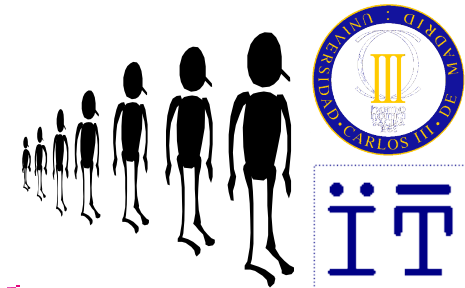
cdk@it.uc3m.es

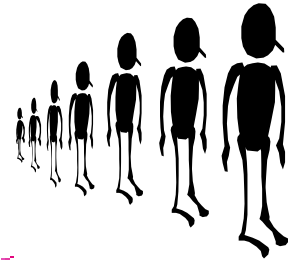
# Recursión en cascada: Fibonacci



```
public static long fib (int n)
{if (n<=1)
    return 1;
else
    return fib (n-1)+fib (n-2) ;
}
```

# Fibonacci

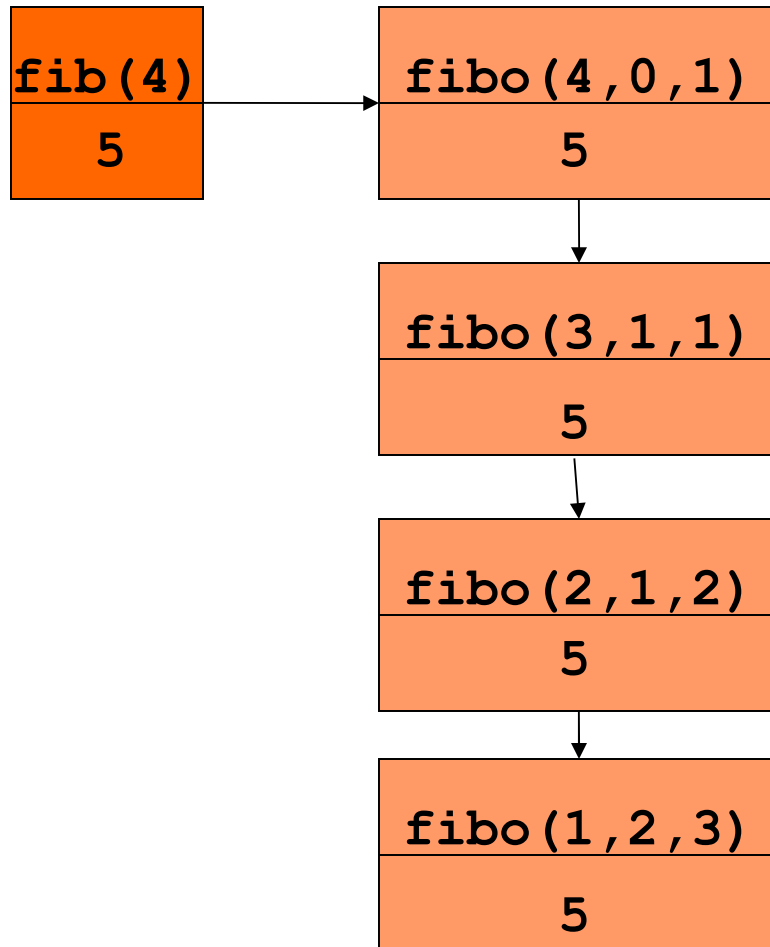
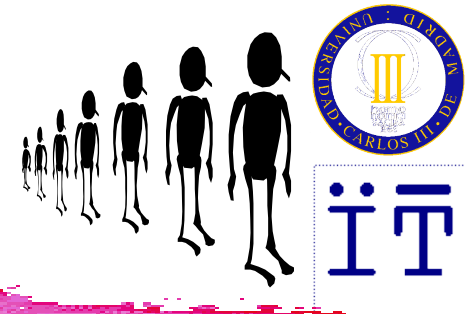




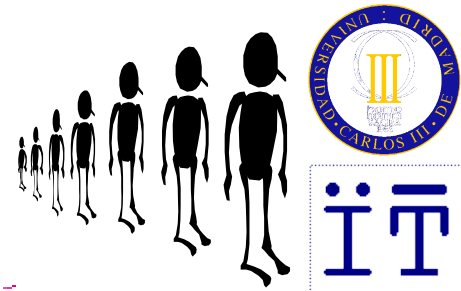
# Fibonacci lineal

- ```
public static long fibo (int n,x,y)
{if (n<=1)
    return x+y;
else
    return fibo(n-1,y,x+y);
}
```
- ```
public static long fib (int n)
{return fibo(n,0,1);}
```

# Fibonacci

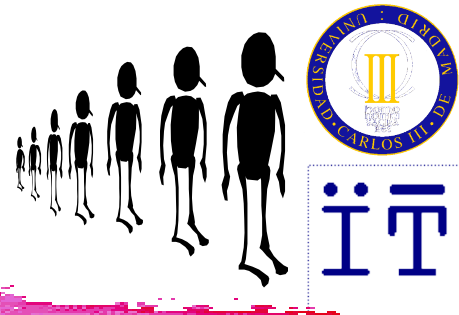


# Fibonacci no recursivo



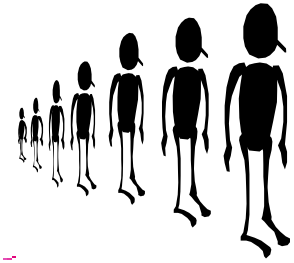
$$\text{fib}(n) = \frac{(1 + \sqrt{5})^{n+1} - (1 - \sqrt{5})^{n+1}}{(2^{n+1} \cdot \sqrt{5})}$$

# Recursión anidada: Morris



```
public static long mor(int n, m)
{if (n==m)
    return (m+1);
else
    return mor(n,mor(n-1,m+1));
}
```

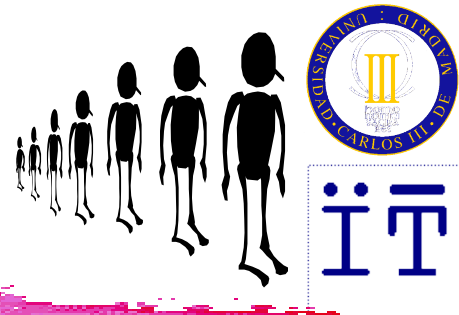




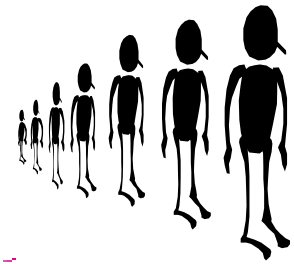
# Morris

- `mor(4, 0) =`
- `mor(4, mor(3, 1)) =`
- `mor(4, mor(3, mor(2, 2))) =`
- `mor(4, mor(3, 3)) =`
- `mor(4, 4) =`
- 5

# Recursión anidada: Ackermann

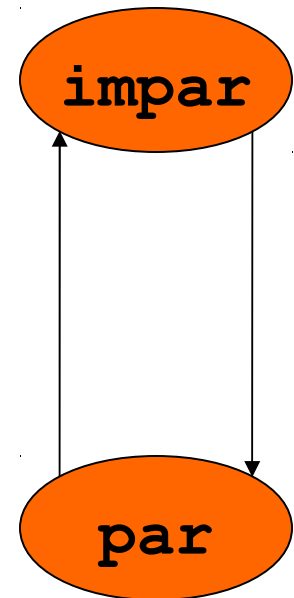


```
public static long ack (int n, m)
{if (n==0)
    return (m+1);
else if (m==0)
    return ack (n-1,1);
else
    return ack (n-1,ack (n,m-1));
}
```

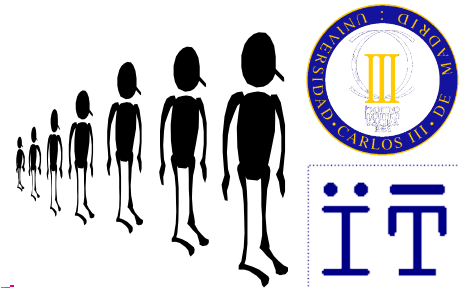


# Recursión mutua

- public static boolean impar (int n)  
{if (n==0)  
    return false;  
else  
    return par(n-1);  
}
- public static boolean par (int n)  
{if (n==0)  
    return true;  
else  
    return impar(n-1);  
}



# Recursión vs. Iteración

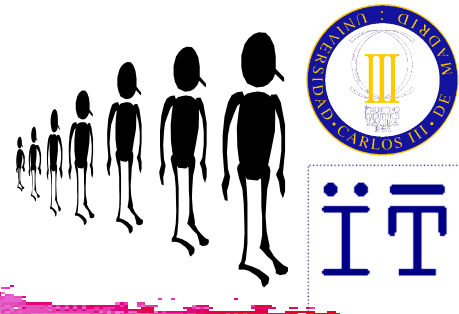


- La recursión por la cola se puede convertir de forma inmediata en iteración (bucle).
- Para otras formas de recursión se requieren técnicas de transformación de programas y posiblemente estructuras de datos más complejas.

*"The transformation from recursion to iteration is one of the most fundamental concepts of computer science." -- D. Knuth 1974*



# Factorial



```
public static long
fact (int n,m)
{if (n<=1)
    return m;
else
    return
    fact (n-1, n*m) ;
}
```

```
public static long
fact (int n,m)
{private int N,M;
 N=n; M=m;
while !(N<=1)
    {M=N*M; N=N-1;}
return M;
}
```