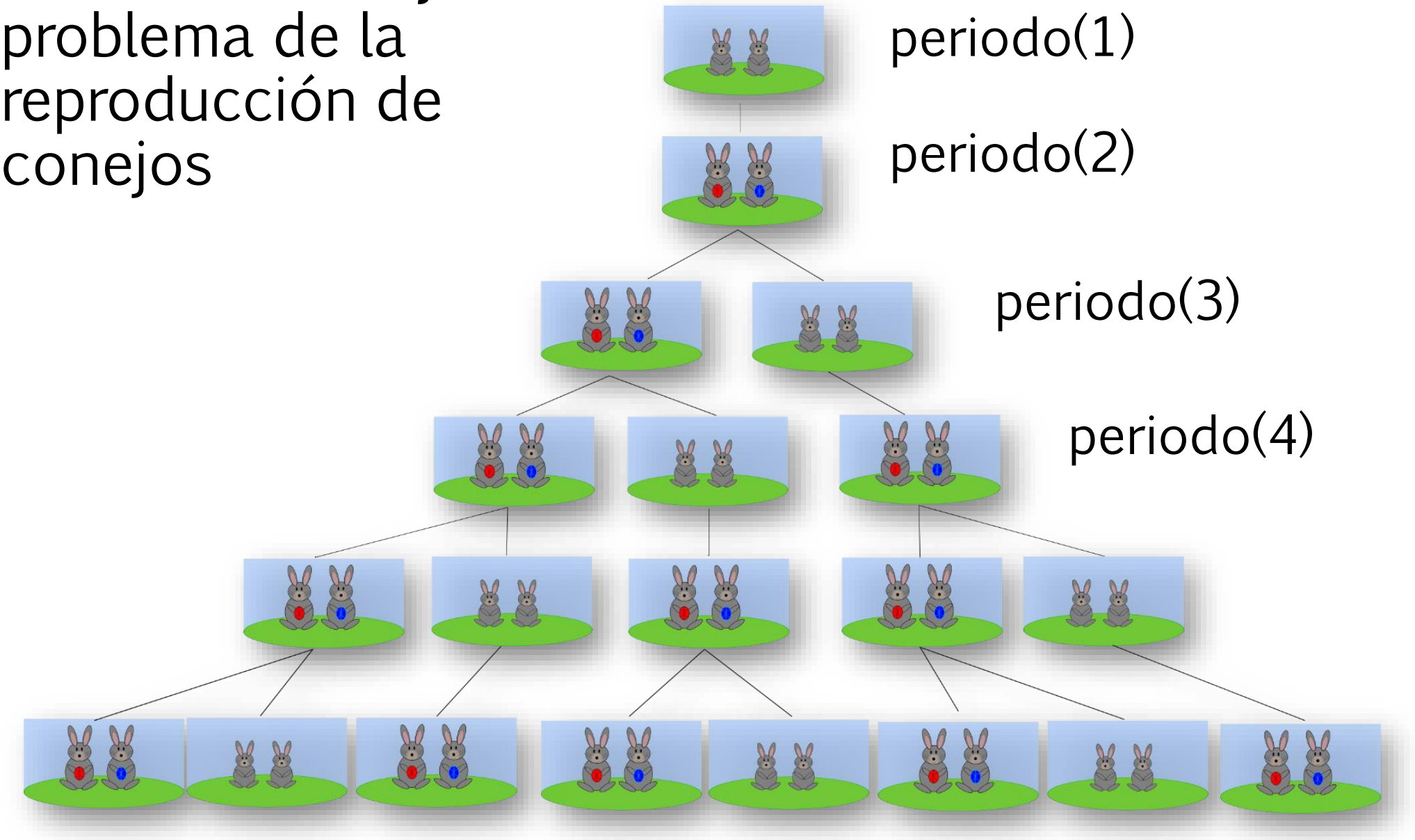
Algorítmica y Programación I

RECURSIÓN

Recursión – ej: problema de la reproducción de conejos



Definiciones

- La recursividad es una herramienta que permite expresar la resolución de problemas evolutivos, donde es posible que un módulo se invoque a si mismo en la solución del problema. (Garland, 1986),(Hellman, 1991), (Aho, 1988)

- Se puede considerar como un caso particular de programación en el que se plantea la resolución en términos de subproblemas más sencillos.

- Cuando se obtienen soluciones a problemas en los que un subprograma se llama así mismo para resolver el problema, se tienen subprogramas recursivos.

Caso de estudio

- Características
  - Existe el **caso base**, aquel que no provoca un nuevo cálculo recursivo
  - Existe un **conjunto de casos recurrentes** que son los que producen un nuevo cálculo recursivo.
- Construcción
  - ¿Cómo se puede definir el problema en términos de un problema mas simple de la misma naturaleza?
  - ¿Cómo será disminuido el tamaño del problema en cada llamado?
  - ¿Qué situación servirá como caso base?

Análisis – Función factorial

› Matemáticamente

factorial(n) = n * (n-1) * (n-2)* …. * 1

factorial(0) = 1

Ej: factorial(4) = 4 * 3 * 2 * 1 = 24

$$n! = \prod_{k=1}^{n} k$$

Análisis – Función factorial

› Estrategia iterativa

```
function factorial(n:byte):longint;
var i:integer; fact:longint;
begin
    fact := 1;
    for i:= 1 to n do
            fact := fact * i;
    factorial := fact;
end;
```

Análisis – Función factorial

› Estrategia recursiva

```
function factorial(n:byte):longint;
begin
    if (n <=1)then
            factorial := 1
    else
            factorial := n * factorial(n-1);
end;
```

FUNCIÓN FACTORIAL MEMORIA – (PILA)

Factorial= ? N=1

Factorial= 2 * ? N=2

Factorial= 3 * ? N=3

Factorial= 4 * ? N=4

factorial(4)

```
function factorial(n):longint
   if n<=1 then factorial := 1
   else factorial := n * factorial(n-1)
```

```
function factorial(n=3):longint
   if n<=1 then factorial := 1
   else factorial := 3 * factorial(2)
```

```
function factorial(n=2):longint
   if n<=1 then factorial := 1
   else factorial := 2 * factorial(1)
```

```
function factorial(n=1):longint
   if n<=1 then factorial := 1
```

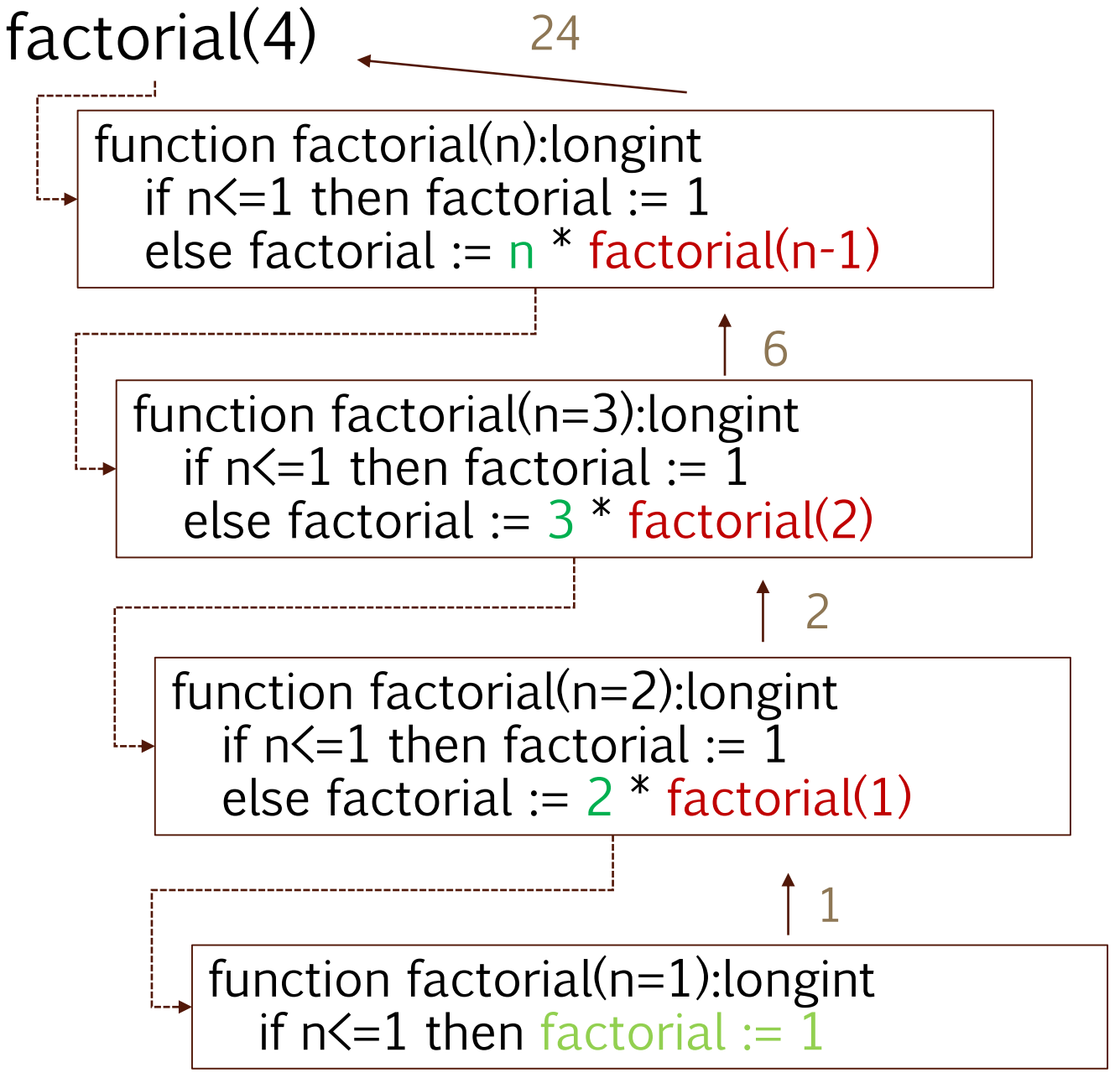
FUNCIÓN FACTORIAL MEMORIA – (PILA)

Factorial= 1 N=1

Factorial= 2 * 1 N=2

Factorial= 3 * 2 N=3

Factorial= 4 * 6 N=4



Conclusiones

- La recursividad es una estrategia poderosa para expresar la solución de forma sintética y clara de aquellos problemas de naturaleza recursiva.
- Es una herramienta que puede ayudar a reducir la complejidad de un programa.
- Las soluciones recursivas son menos eficientes que las iterativas, dado el consumo de tiempo y memoria.

Ejercicios

- Calcular la potencia n-esima de un numero i (i^n)

- Leer n caracteres e imprimirlos en orden inverso.

- Calcular la función de fibonacci.
  - f(0) = 0
  - f(1) = 1

  Si (n>1) :: $f(n) = f(n-1) + f(n-2)$

- Simular el movimientos de las Torres de Hanoi.

Bibliografia

- Aho, Hopcroft, Ullman “Estructuras de datos y algoritmos”. Addison Wesley, 1998.
- Armando E. de Gusti “Algoritmos Datos y programas”. Pearson Education 2001.
- Recursos gráfico
  - https://commons.wikimedia.org/wiki/File:FibonacciRabbit.svg