RECURSIVIDAD

Ing. Edgar Gerardo Salinas Gurrión

Recursividad

- Una función que se llama así misma, ya sea directa o indirectamente, se denomina recursiva.
- Utilidad: cuando la solución de un problema se puede expresar de términos de la resolución de un problema de la misma naturaleza, aunque de menor complejidad.

Recursividad

- Divide y vencerás: un problema se divide en otros problemas más sencillos (del mismo tipo).
- Sólo tenemos que 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 bases que hayamos definido.

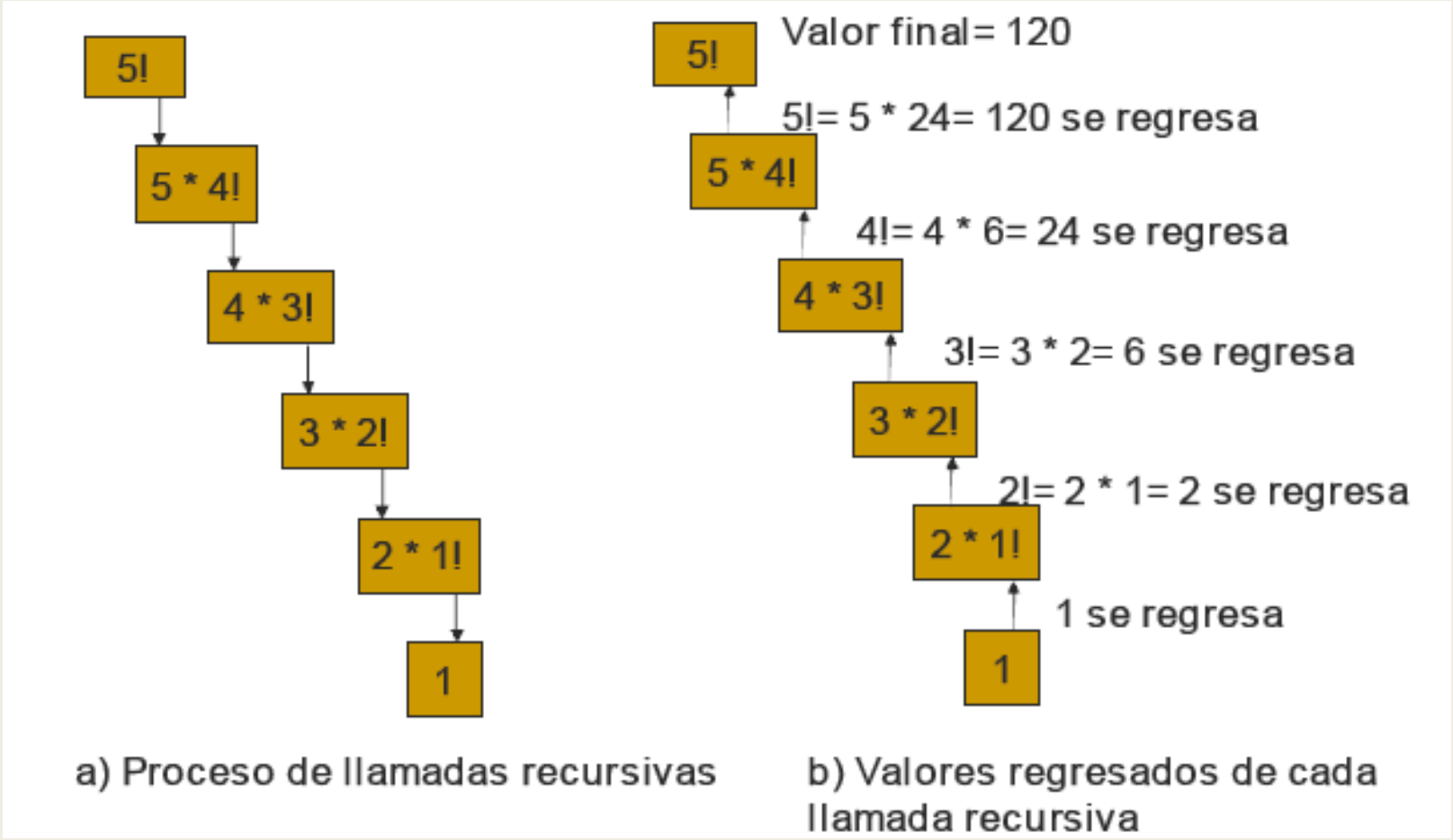
Recursividad

- La llamadas recursivas simplifican el problema y, en última instancia, los casos bases nos sirven para obtener la solución.
- El proceso de recursividad puede incluir más llamadas a métodos recursivos.

Método recursivo para el factorial

```
public long factorial ( long number ) {
    if( number<=1 )
        return 1;
    else
        return number * factorial ( number – 1 );
}
```

Evaluación recursiva de 5!



Diseño de algoritmos recursivos

1. Resolución del problema para los casos base:
   - *Sin emplear recursividad.*
   - *Siempre debe de existir algún caso base.*

2. Solución para el caso general:
   - *Expresión de forma recursiva.*
   - *Pueden incluirse pasos adicionales (para combinar las soluciones parciales)*
   - *Debe progresar hacia el caso base*

Recursividad vs. Iteración

- Tanto la recursividad como la iteración están basadas en estructuras de control:
- La iteración usa estructuras de repetición (for, while o do while);
- La recursividad usa estructura de selección (if, if/else o switch).

Recursividad vs. Iteración

- Cualquier problema que se pueda resolver mediante la recursividad puede resolverse mediante la iteración.
- Aspectos que hay que considerar al decidir cómo implementar la solución a un problema:
  - *La carga computacional (tiempo en CPU y espacio en memoria) asociada a las llamadas recursivas.*
  - *Normalmente la solución recursiva es menos eficiente que la iterativa.*

Recursividad vs. Iteración

- *La redundancia (algunas soluciones recursivas resuelven el mismo problema en repetidas ocasiones).*
- *La complejidad de la solución (en ocasiones, la solución iterativa es muy difícil de encontrar).*
- *La concisión, legibilidad y elegancia del código resultante (la solución recursiva del problema puede ser más sencilla).*

Método recursivo para calcular la serie de Fibonacci

```
public long fibonacci ( long n ) {
    if( n == 0 || n == 1)
        return n;
    else
        return fibonacci ( n – 1 ) + fibonacci( n – 2 );
}
```

Método iterativo para calcular la serie de Fibonacci

```
static int fibonacci ( int n ) {
    int actual, ant1, ant2;
    ant1= ant2 + 1;
    if( n == 0 || n == 1)
        actual= 1;
    else
    for (i=2; i<=n; i++){
        actual= ant1 + ant2;
        ant2=ant1;
        ant1= actual;
    }
}
```