Tema 11:
Recursividad

Indice

1. **Introducción**
2. **Funcionamiento de una función recursiva**
3. **Ejemplos**

   Algoritmo de ordenación quicksort

1. Introducción

- La recursividad es el proceso de definir algo en términos de sí mismo (definición circular)
- Un lenguaje de programación permite recursividad cuando una función puede llamarse a sí misma

$$\text{Factorial}(n) \begin{cases} 1 & \text{si } n=1 \\ n * \text{Factorial}(n-1) & \text{en caso contrario} \end{cases}$$

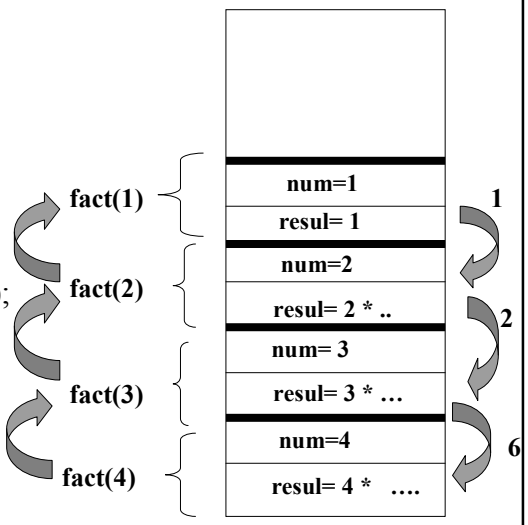
- En toda definición recursiva debe existir un caso base y un caso recursivo
- Además el caso recursivo debe acercarse o "converger" en el caso base.

2. Funcionamiento

- Cuando se realiza una llamada a una función se le asigna un espacio en la zona de memoria (pila) para las nuevas variables locales y parámetros.
- Cuando las llamadas son recursivas el código de la función se ejecuta desde el principio con los nuevos valores de los parámetros.
- Al volver de la llamada recursiva se recuperan las variables locales y los parámetros antiguos y se reanuda la ejecución por donde iba.

Funcionamiento de la pila

```
int fact(int num )
{
    int resul;
   if ( num==1)
      resul=1;
   else
      resul= num * fact (num-1);
  return resul;
}
```



Recursivo versus Iterativo

- Para cada solución iterativa del problema existe una solución recursiva y viceversa.
- Dependiendo del problema será mejor la solución recursiva o iterativa.
- En caso de arrays, la recursividad suele basarse en el tamaño del array.
- La principal ventaja de soluciones recursivas es que para determinados problemas aportan soluciones más claras y sencillas que sus equivalentes iterativas.

3. Ejemplos

- Calcular la potencia de un número b elevado a un exponente e (ambos positivos)
- CASO base

  e=1 potencia (b, e) = b
- CASO RECURSIVO

  potencia(b,e) = b* potencia (b, e-1)

Algoritmo de ordenación quicksort

- Es una algoritmo recursivo de ordenación de los elementos de un vector
- Se basa en
  - Elección de un elemento pivote
  - Realizar intercambios para poner el pivote en su posición correcta. Además todos los elementos a la derecha del pivote son mayores que él y todos los de la izquierda son menores que él (si ordenamos de forma creciente).
  - Ver quicksort.cpp