



Computación 1

Curso 2024

Facultad de Ingeniería
Universidad de la República



Introducción

Definiciones (según RAE)

■ Recursión

- Sinónimo de recursividad

■ Recursividad

- Cualidad de recursivo

■ Recursivo

- Que se contiene a si mismo un número indefinido de veces.
- Dicho de una unidad o una estructura: Que puede contener como constituyente otra del mismo tipo.

Introducción

Definiciones

■ Recursión

- ❑ Técnica algorítmica donde el algoritmo se llama a sí mismo para realizar una tarea.
- ❑ Un algoritmo es recursivo si se define en términos de sí mismo.

Introducción

Definiciones

■ Recursión en Matemáticas

- El concepto de recursión es una herramienta básica
- Principio de Inducción Completa
- Definición de Conjuntos:
 - Números naturales:
 - 1 es un número natural
 - El siguiente de un número natural es un número natural
- Definición de Funciones
 - La función factorial, $n!$
 - $0! = 1$
 - Si $n > 0$ entonces $n! = n * (n - 1)!$

Introducción

Definiciones

■ Recursión en Programación

- Técnica utilizada en lugar de la iteración cuando:
 - Solución compleja utilizando iteración
 - Solución poco clara al utilizar iteración
- Problemas cuya solución se puede hallar resolviendo el mismo problema pero con un caso de menor tamaño.



Introducción

Definiciones

- Programa Directamente Recursivo

- Se llama a sí mismo dentro de su cuerpo de sentencias

- Programa Indirectamente Recursivo

- En su cuerpo de sentencias posee una invocación a otro programa que lo invoca nuevamente

Programa Recursivo

Definición

- **¿Cómo resuelve un problema?:**
 - Se llama a si mismo con una versión más simple del problema ... y hace algo extra para completar la resolución.
- **Tiene uno o varios Casos Recursivos:**
 - donde se llama a si mismo. Resuelve los casos genéricos.
- **Tiene uno o varios Casos Base:**
 - Resuelve los casos evidentes que no requieren recursión.

Programas Recursivos

Metodología

- Resolución de los casos simples (casos base).
 - Encontrar y resolver los casos de resolución simple, sin necesidad de recurrencia.
 - Condición de Parada o Salida
- Resolución del caso general utilizando casos más pequeños.
 - Llamada recursiva.
 - Algún dato de entrada siempre está más cerca del caso base, garantizando terminación.

Programas Recursivos

Metodología

- Combinar las soluciones de los distintos casos conformando la solución al problema.
 - Utilizar estructuras de selección.
 - if ... elseif ... else ... endif
 - switch ... case otherwise ... endswitch

Programas Recursivos

Metodología

■ Definición

- `factorial(0) = 1`

- `factorial(n) = n*factorial(n-1)`

■ Caso Base

- Si $n = 0$ entonces `factorial(n) = 1`

- No es necesario recurrir

■ Caso Recursivo o General

- Si $n > 0$ entonces `factorial(n) = n*factorial(n-1)`

- Se recurre al factorial del natural anterior

Programas Recursivos

Metodología

```
function fn=factorial(n)
```

```
if n==0  
    fn=1;
```

Caso Base o Condición de Salida

```
else
```

```
    fn=n*factorial(n-1);
```

Caso Recursivo

```
end
```

Selección

Caso más pequeño

Programas Recursivos

Errores comunes

- Ausencia del caso base.
 - Caso base = Condición de Parada.
 - Si falta, nunca termina la ejecución.
 - Utilizar estructuras de selección para garantizar la ejecución del caso base.
- Error en la llamada recursiva
 - Cada llamada recursiva se realiza con un valor de parámetro que hace el problema “de menor tamaño”.

Programas Recursivos

Errores comunes

- Utilizar estructuras iterativas en lugar de estructuras selectivas.
 - La llamada recursiva se realiza en una sentencia de selección.

Programas Recursivos

Ejecución

■ Cálculo de factorial(3)

$$\text{factorial}(3) = 3 * \text{factorial}(2)$$

$$\text{factorial}(2) = 2 * \text{factorial}(1)$$

$$\text{factorial}(1) = 1 * \text{factorial}(0)$$

$$\text{factorial}(0) = 1 \text{ Caso Base}$$

$$\text{factorial}(1) = 1 * 1$$

$$\text{factorial}(2) = 2 * 1$$

$$\text{factorial}(3) = 3 * 2$$

Programas Recursivos

Utilización de Memoria

- Cada paso de la recursión ejecuta una nueva versión del programa,
- Nueva asignación de espacio de memoria en cada paso.

Programas Recursivos

Redundancia

■ Función Fibonacci

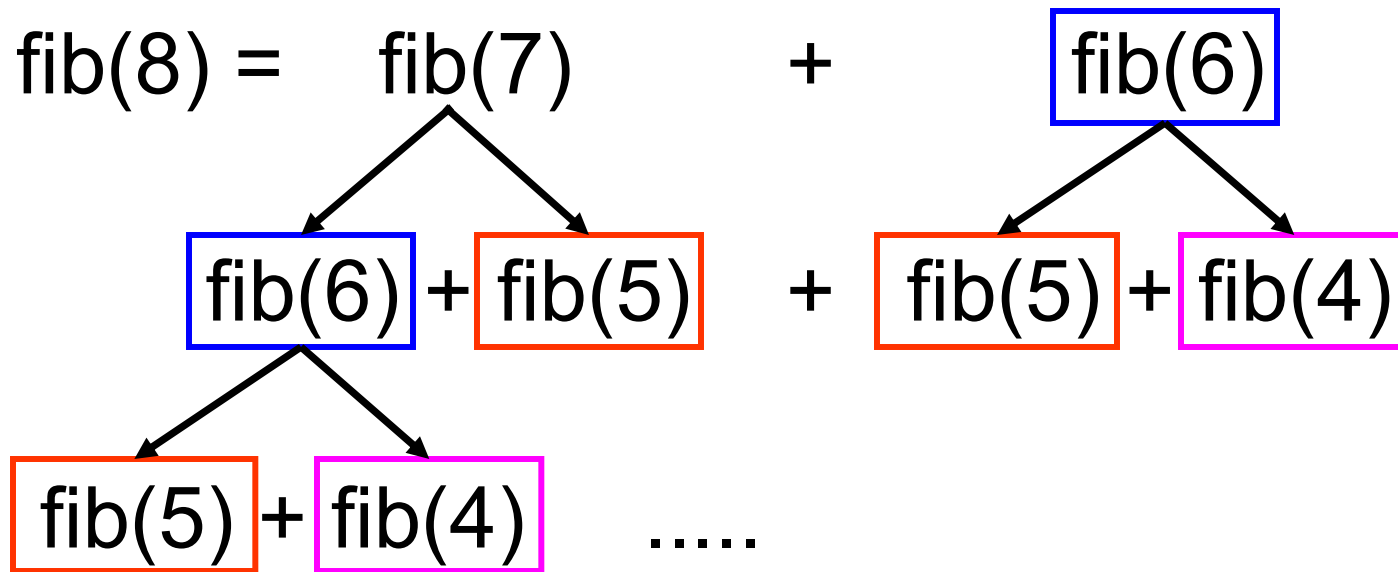
- `Fib(1) = 1`
- `Fib(2) = 2`
- `Fib(n) = Fib(n-1) + Fib(n-2)`

■ Programa Recursivo

```
function fn=fib(n)
if n == 1
    fn = 1;
elseif n == 2
    fn = 2;
else
    fn = fib(n-1) + fib(n-2);
end
```

Programas Recursivos

Redundancia



■ Redundancia en cálculos