Violeta Ocegueda, Leocundo Aguilar, y Mauricio Sánchez
Facultad de Ciencias Químicas e Ingeniería
Universidad Autónoma de Baja California

# AED

## Algoritmos y Estructuras de datos

# Recursión

## Definición y características

- Una función recursiva es aquella que se invoca a sí misma.
- Permite expresar algoritmos complejos en forma compacta y elegante sin reducir la eficiencia.
- Un algoritmo recursivo es aquel que resuelve un problema resolviendo una o más instancias menores que el mismo problema.

# Recursión

## Reglas de la recursión

1. **Caso base:** Siempre debe existir casos base que se resuelven sin hacer uso de la recursión.
2. **Caso recursivo:** Cualquier llamada recursiva debe progresar hacia un caso base.
3. **Diseño:** Asumir que toda llamada recursiva interna funciona correctamente.
4. **Regla de Interés compuesto:** Evitar duplicar el trabajo resolviendo la misma instancia de un problema en llamadas recursivas compuestas.

# Recursión

## **Ejemplo1:** Calcular el factorial de n.

- El factorial de un entero positivo n es el producto de todos los números enteros positivos desde 1 hasta n.
- $n! = 1 \times 2 \times 3 \times 4 \times ... \times (n-1) \times n$
- También se le define como:

$$n! = \begin{cases} 1 & : si, n = 0 \\ (n-1)! \times n & : si, n > 0 \end{cases}$$

- $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120 \leftrightarrow 5! = 5 \times 4!$
- $4! = 4 \times 3 \times 2 \times 1 \leftrightarrow 4! = 4 \times 3!$
- $3! = 3 \times 2 \times 1 \leftrightarrow 3! = 3 \times 2!$
- $2! = 2 \times 1 \leftrightarrow 2! = 2 \times 1!$
- $1! = 1$

# Recursión

**Ejemplo 1:** Calcular el factorial de n.

```
int factorial(int n)
{
      if (n <= 1)
          return 1; // Caso base
      else
          return n * factorial(n - 1); // Caso recursivo
}
```

Donde para n = 3

- $3 * factorial(3 - 1)$
- $2 * factorial(2 - 1)$
- $1$
- $1 * 2 * 3 = 6$

# Recursión

**Ejemplo 2:** Calcular el n-ésimo término de la sucesión Fibonacci.

- Comienza con los números $0$ y $1$, y a partir de estos, cada término es la suma de los dos anteriores.
- $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, \ldots$
- La sucesión se puede definir por las siguientes ecuaciones:
  1. $f_0 = 0$
  2. $f_1 = 1$
  3. $f_n = f_{n-1} + f_{n-2}$
- De tal manera que:
  - $f_2 = f_1 + f_0 = 1 + 0 = 1$
  - $f_3 = f_2 + f_1 = 1 + 1 = 2$
  - $f_4 = f_3 + f_2 = 2 + 1 = 3$
  - $f_5 = f_4 + f_3 = 3 + 2 = 5$
  - $f_6 = f_5 + f_4 = 5 + 3 = 8$

# Recursión

**Ejemplo 2:** Calcular el n-ésimo término de la sucesión Fibonacci.

```
int fibonacci(int n)
{
    if (n == 0 || n == 1)
        return n; // Caso base
    else
        return fibonacci(n-1) + fibonacci(n-2); // Caso
            recursivo
}
```

Donde para n = 5 (quinto término)

- $fibonacci(5) = fibonacci(4) + fibonacci(3)$
- $fibonacci(4) = fibonacci(3) + fibonacci(2)$
- $fibonacci(3) = fibonacci(2) + fibonacci(1)$
- $fibonacci(2) = fibonacci(1) + fibonacci(0)$
- $fibonacci(1) = 1$
- $fibonacci(0) = 0$

# Recursión

**Ejercicios:**

- Escribir una función que devuelva la suma de los primeros N enteros positivos.
- Escribir una función que devuela la suma de los enteros positivos pares desde N hasta 2, donde N es un número par.
- Escribir una función que transforme un número entero positivo a notación binaria.
- Escribir una función que calcule el máximo común divisor de dos enteros positivos.

# Gracias