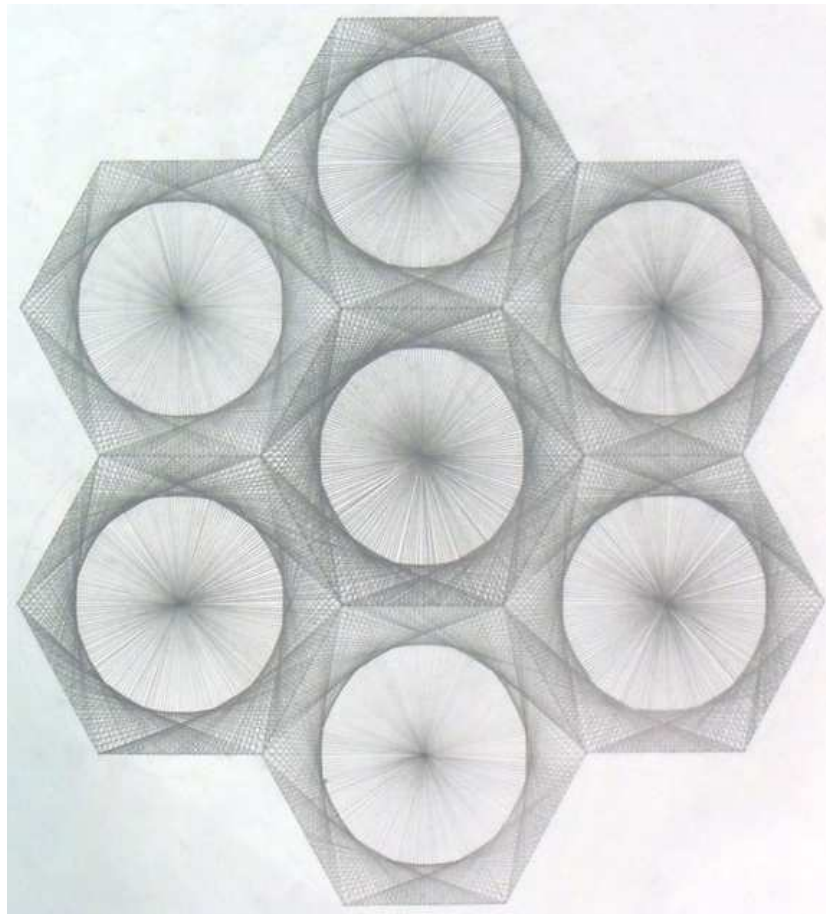


PRÁCTICA 6. RECURSIVIDAD

OBJETIVOS:

- ✓ Comprender el concepto de módulo recursivo
- ✓ Comprender la ejecución de un módulo recursivo mediante la realización de trazas
- ✓ Utilizar métodos recursivos para la resolución de problemas que pueden definirse de modo natural en términos recursivos, tales como muchas funciones matemáticas



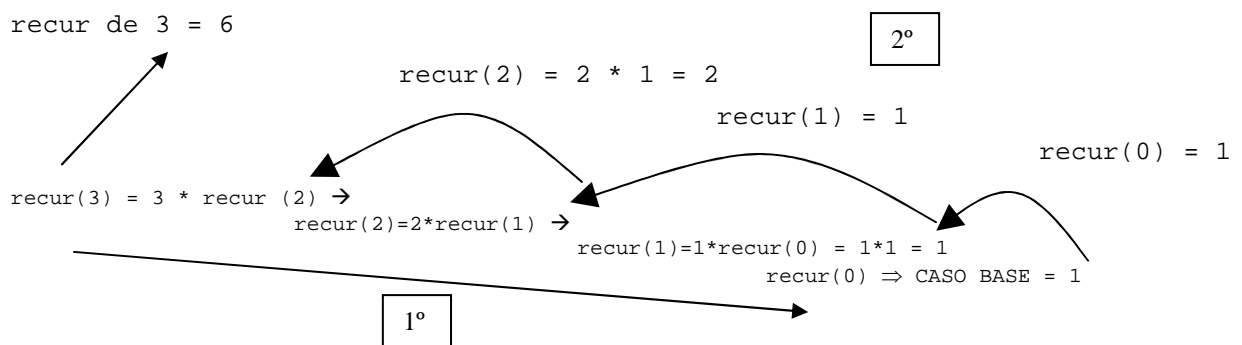
Ejercicio Resuelto 1. Realiza varias trazas de la siguiente función recursiva y describe en lenguaje natural qué calcula la función.

```
int recur(int n)
{
    int result;
    if (n==0)
        result = 1;
    else
        result = n * recur(n - 1);

    return(result);
}
```

En esta traza :

1. Se producen llamadas para calcular **recur(n)** con n decreciendo sucesivamente. La resolución de **recur(n)** se queda en espera, hasta que se calcule **recur(n - 1)**
2. Se llega a un caso base, tras el cual se produce una vuelta atrás en la secuencia anterior, para terminar la resolución de las distintas llamadas que se habían quedado en espera.



Esta función recursiva calcula el factorial de un número.

Ejercicio Resuelto 2. Realiza un programa en C que pida por teclado n números enteros y los muestre por pantalla (recursivamente) en sentido contrario al leído.

Ejemplo:

```
Introduce número: 34
Introduce número: 7
Introduce número: 102
Introduce número: 65
65
102
7
34
```

El procedimiento recursivo se puede implementar de la siguiente forma:

```
void invertir(int n) // n es el total de números a introducir
{
    int num;

    if(n>0)
```

```

{
    cout << "Introduce número: ";
    cin >> num;

    invertir(n-1);
    cout << num << endl;
}
}

```

¿Qué sucedería si hacemos la llamada recursiva después de imprimir el número por pantalla?

Ejercicio Resuelto 3. Implementa un módulo recursivo en C que, dado un número en decimal, imprima por pantalla su correspondiente binario.

Está basado en el método que usamos para convertir un número decimal a binario “a mano”: dividir el número entre dos hasta el caso base e imprimir los restos del último al primero:

```

void decimalABinario(int n)
{
    if(n<=1)
        cout << n;
    else
    {
        decimalABinario(n/2);
        cout << n%2;
    }
}

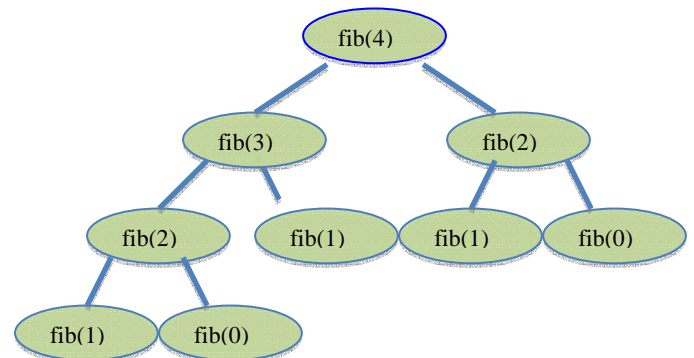
```

Ejercicio Resuelto 4. Implementa en C un programa recursivo que calcule el valor de la serie de Fibonacci para un número n dado. Recuerda que la serie de Fibonacci es la siguiente: 1 1 2 3 5 8 13...

Árbol de llamadas recursivas para fib(4):

Su definición matemática es:

$$\text{fib}(n) = \begin{cases} \text{fib}(n) = 1, \text{ si } n=0 \text{ ó } n=1 \\ \text{fib}(n) = \text{fib}(n-2) + \text{fib}(n-1) \end{cases}$$



La función recursiva que calcula esta fórmula se implementa de la siguiente forma:

```

int fib(int n)
{
    int resultado;

    if(n == 0 || n == 1)
        resultado = 1;
    else
        resultado = fib(n-1) + fib(n-2);

    return resultado;
}

```

Ejercicio Resuelto 5. **Dada la siguiente función en C que devuelve la suma de los dígitos del número pasado como argumento, implementa una función equivalente recursiva:**

```
int sumaDigitos(int n)
{
    int res = 0;

    while(n>9)
    {
        res += n%10;
        n = n / 10;
    }
    res += n;
    return res;
}
```

Su equivalente recursivo es:

```
int sumaDigitosRecur(int n)
{
    int res;

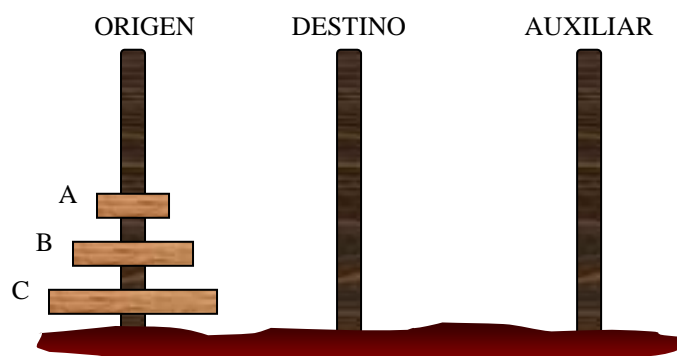
    if (n < 10)
        res = n;
    else
        res = n%10 + sumaDigitosRecur(n/10);

    return(res);
}
```

Curiosidad: El problema de las torres de Hanoi

Durante algunos siglos los habitantes de un monasterio en Hanoi no tenían apenas relación con el mundo exterior y ello les llevó a idear algún modo de entretenimiento. Construyeron tres largos postes que colocaron en el jardín y otros se encargaron de construir distintos discos de madera de varios tamaños. El entretenimiento les llevaba varias horas al día y bastantes meses, pues no conocían la estrategia adecuada para finalizar rápidamente.

Hemos pensado que podrías ayudarles con un sencillo algoritmo recursivo que resolviese su juego. ¿Probamos? El juego de las torres de Hanoi consiste en tener unos discos colocados del siguiente modo:



Dichos discos están perforados y se trata de pasarlos al poste llamado DESTINO, pero de forma que en ningún momento quede un disco colocado sobre otro de tamaño inferior, para lo que se puede emplear temporalmente el poste AUXILIAR.

```
#include <iostream>

using namespace std;

void transferir(int,char,char,char);

int main(void) {
    int n;
    cout << "Bienvenido a las torres de Hanoi"<<endl;
    cout << "Cuántos discos?";
    cin >> n;
    transferir(n,'I','D','C');
}

// La función devuelve un número entero
void transferir(int discos, char desde, char hacia, char temp)
{
    if (discos>0)
    {
        transferir(discos-1,desde,temp,hacia);
        cout << "Mover disco '"<<discos<<"' desde "<<desde<<" hasta "<<hacia<<endl;
        transferir(discos-1,temp,hacia,desde);
    }
}
```