Práctica 6. Recursividad

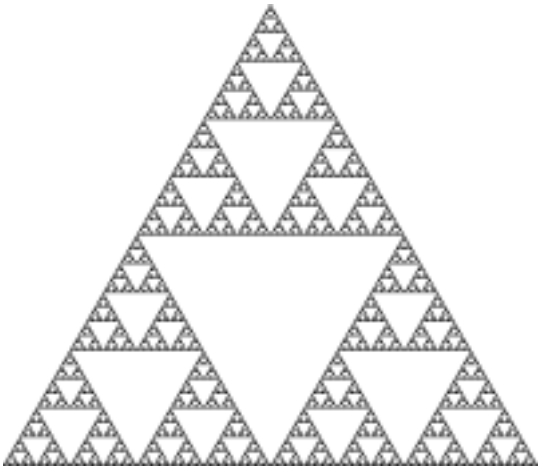
Objetivos:

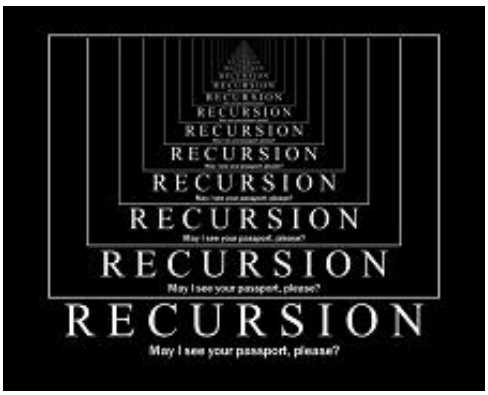
- Comprender el concepto de módulo recursivo.
- Comprender la ejecución de un módulo recursivo mediante la realización de trazas.
- Utilizar métodos recursivos para la resolución de problemas que pueden definirse de modo natural en términos recursivos, tales como muchas funciones matemáticas.

1. Recursividad

Un módulo es recursivo cuando entre la lista de instrucciones que lo forman, se encuentra una **llamada a sí mismo**, directa o indirectamente. Muchas funciones matemáticas se definen de forma natural de manera recursiva como el factorial de un número o la potencia.

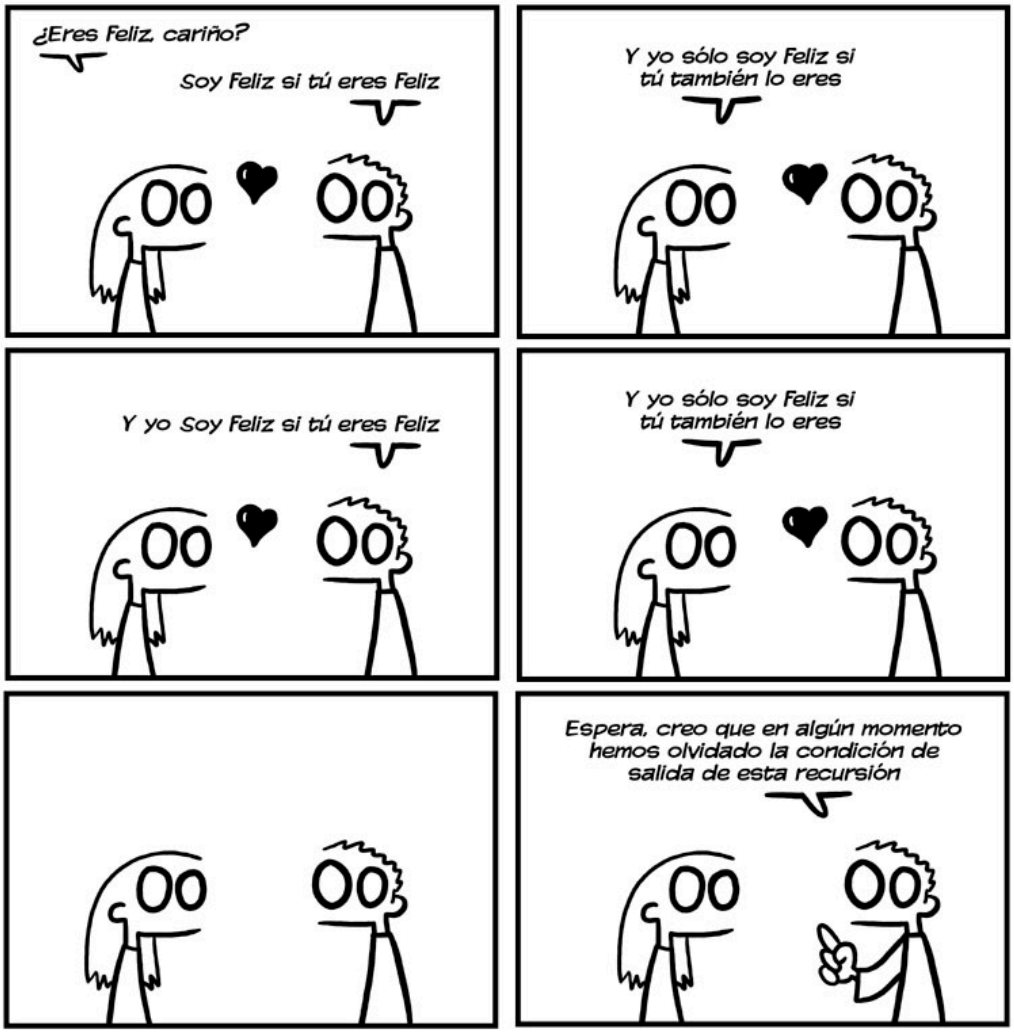
Ejemplo de imágenes recursivas:





Recordad que un módulo recursivo se debe componer de:

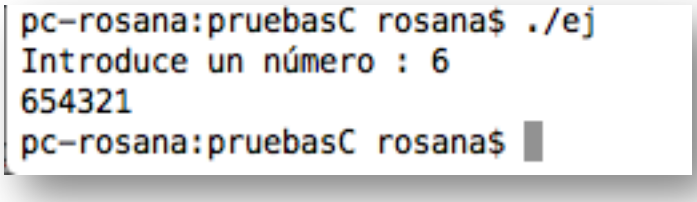
- Uno o más casos base, donde no hay llamadas recursivas y que especifican la condición de terminación de la recursión.
- Uno o más casos generales o recursivos, donde se encuentran llamadas al propio módulo. Estas llamadas deben resolver versiones "más pequeñas" de la tarea inicial. Es decir, tiene que haber una tendencia hacia el caso base.



2. Ejercicio resuelto 1. Repasa

▶ **Diseña una función recursiva que a partir de un número dado muestra por pantalla todos los números naturales desde ese número hasta el 1, es decir, en orden decreciente: n, n-1, n-2..., 2, 1**

Un ejemplo de ejecución sería:



2.1 Solución

```
#include<iostream>
using namespace std;

void alreves(int n) {
   if  (n==1)
      cout << n;
   else {
      cout << n;
      alreves(n-1);
   }
}

main() {
      int num;

      cout << "Introduce un número : ";
      cin >> num ;
      alreves(num);
      cout << endl;
}
```

3. Ejercicio propuesto 1. Resuelve

▶ **Modifica el ejercicio anterior de modo que imprima los números en orden creciente desde 1 hasta el número dado, esto es, 1, 2, ... n**

4. Ejercicio resuelto 2. Repasa

▶ **A continuación puedes ver el código de una función iterativa para mostrar por pantalla el inverso de un número. Diseña un algoritmo recursivo que realice la misma tarea.**

```
1   void inverso(int n) {
2       int aux;
3
4       while(n>0) {
5           aux = n%10;
6           cout<<aux;
7           n = n/10;
8       }
9   }
```

4.1 Solución

```
1    #include<iostream>
2    using namespace std;
3
4    void inverso(int);
5
6    main(){
7      int num;
8
9      cout<<"Introduzca un número: ";
10     cin >> num;
11
12     cout<<"/n El número inverso es: ";
13     inverso(num);  //llamada a la función recursiva
14     cout<<endl;
15   }
16
17   void inverso(int n){
18     int aux;
19
20     if(n>0)
21     {
```

```
22          aux = n%10;
23          cout<<aux;
24          inverso(n/10);
25      }
26  }
```

5. Ejercicio resuelto 3. Repasa

▶ **Realiza varias trazas de la siguiente función recursiva y describe en lenguaje natural qué calcula la función.**

```
1   int  funcion(int n) {
2       int result;
3
4       if (n==0)
5            result = 1;
6       else
7            result = n * funcion (n - 1);
8       return(result);
9   }
```

5.1 Solución

Esta función calcula el factorial de un número. Tiene en cuenta que:

- El factorial de 0 es 1.

$$0! = 1$$

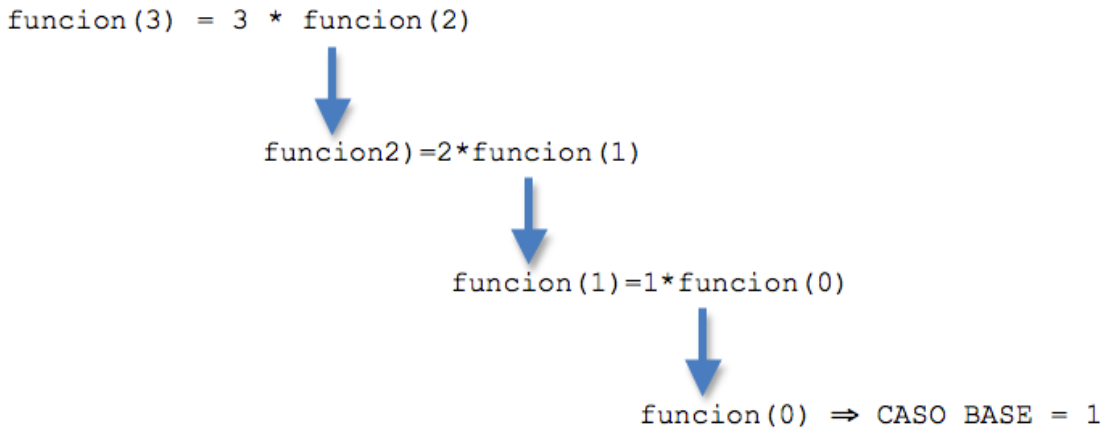
- El factorial de un número 'n', es igual al producto de 'n' por el factorial de 'n-1'.

$$n! = n! * (n-1)!$$

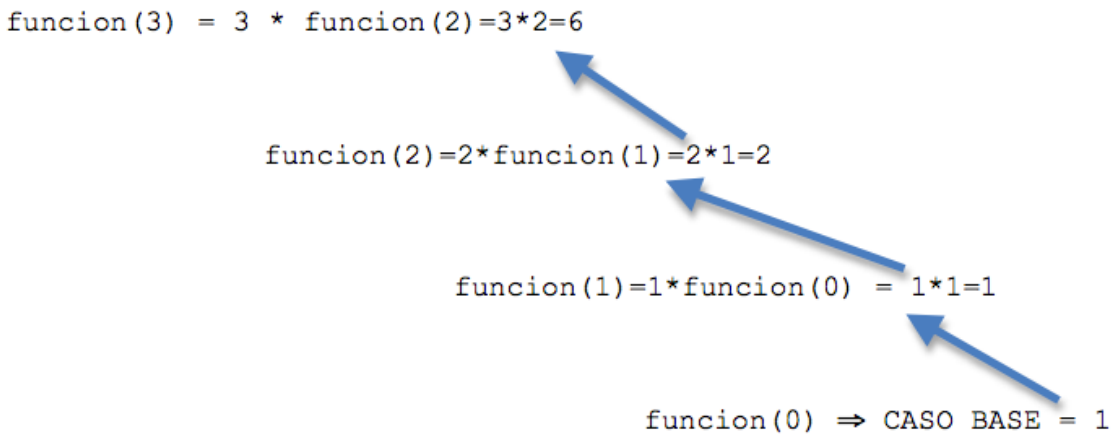
El factorial de 0 es el caso base de la función.

Para realizar la traza nos tenemos que fijar que:

- Se producen llamadas a *funcion(n)* con 'n' decreciendo sucesivamente (acercándose al caso base). La resolución de esa llamada *funcion(n)* se queda en espera hasta que se resuelva *funcion(n-1)*.



- Cuando se llega al caso base, se produce una vuelta atrás en la secuencia de llamadas para determinar el resultado de las diferentes llamadas que se habían quedado en espera.



6. Ejercicio resuelto 4. Repasa

▶ **Realiza un programa en C que calcule de manera recursiva la cifra i-ésima de un número entero. Realiza una traza del problema.**

Ejemplo de ejecución:

```
pc-rosana:pruebasC rosana$ ./ej
Introduce un número: 3758
Introduce la posición que quieres obtener: 5
La posición no es válida
Introduce la posición que quieres obtener: 2
La cifra es: 5
```

6.1. Solución

```
#include <iostream>
#include <cmath>
using namespace std;

int cifra_iesima(int, int);

main() {
	int numero;
	int pos;

	cout << "Introduce un número: ";
	cin >> numero;
	cout << "Introduce la posición que quieres obtener: ";
	cin >> pos;
	//La posición no es válida si es 0 o negativo o si es mayor que la cantidad de cifras que tiene el número
	while (pos<1 || numero/(int)pow(10.0,pos-1)==0) {
		cout << "La posición no es válida" << endl;
		cout << "Introduce la posición que quieres obtener: ";
		cin >> pos;
	}
	cout << "La cifra es: " << cifra_iesima(numero, pos) << endl;
}

int cifra_iesima(int num, int pos) {
	if (pos==1)
		return num%10;
	else
		return cifra_iesima(num/10, pos-1);
}
```

7. Ejercicio propuesto 2. Resuelve

▶ **Tenemos la siguiente función:**

```
int recursiva (int n) {
     int r, c, devolver;

     if (n < 10)
       Devolver = n;
     else      {
         c = n / 10;
         r =  n % 10;
         devolver=recursiva ( c ) + r ;
       }
       return(devolver);
}
```

a) Realiza varias trazas de la función indicando la secuencia de llamadas y sus valores de retorno.

b) Describe en lenguaje natural qué hace dicha función.

c) Escribe un programa en C completo para probar la función.

8. Ejercicio propuesto 3. Resuelve

▶ **Explica qué hace el siguiente ejercicio iterativo y a continuación resuelve el mismo problema de forma recursiva.**

```
int ejercicio(double b, int e){
       double p;

       p = 1;
       for (int i=1; i<= e; i++)
              p = p * b;
       return(p);
 }
```

9. Ejercicio propuesto 4. Resuelve

▶ **Implementa en C un programa recursivo que usando asteriscos dibuje un triángulo en pantalla del tamaño que introducirá el usuario por teclado.**

Fíjate en que dibujar recursivamente un triángulo de tamaño n es lo mismo que dibujar un triángulo de tamaño (n-1) y luego escribir una fila de n asteriscos (la fila la puedes hacer simplemente con un bucle).

Ejemplo de ejecución:

```
pc-rosana:pruebasC rosana$ ./ej
Introduce el número de filas: 6
*
**
***
****
*****
******
pc-rosana:pruebasC rosana$
```

Una vez implementado lo anterior, prueba a hacer ahora el dibujo de un triángulo invertido, también de manera recursiva.

Ejemplo de ejecución:

```
pc-rosana:pruebasC rosana$ ./ej
Introduce el número de filas: 6
******
*****
****
***
**
*
pc-rosana:pruebasC rosana$
```

10. Ejercicio propuesto 5. Resuelve

▶ **Implementa la función recursiva** `int sumaMult5(int n)` **que devuelva la suma de los múltiplos de 5 que hay desde 1 hasta n. Por ejemplo: si n es el número 22 devolverá 50 (ya que los múltiplos de 5 que hay hasta 22 son 5, 10, 15 y 20 y su suma es 50).**

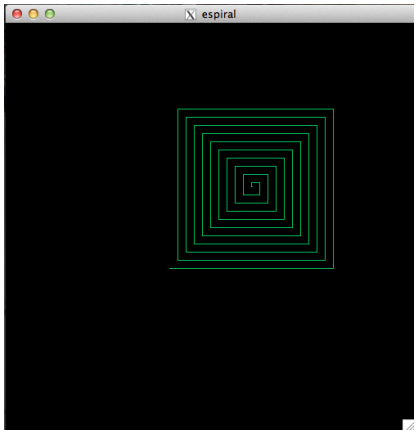
11. Ejercicio propuesto 6. Resuelve

- **Tomando como base el ejercicio propuesto 5 de la práctica 3 (el ejercicio gráfico), haz otro programa (espiral.c) que dibuje de manera recursiva una espiral a derechas. Piensa que dibujar una espiral no es más que dibujar en línea recta un determinado número de píxeles, torcer a la derecha 90 grados y luego dibujar una espiral más pequeña (del tamaño inicial menos una determinada cantidad). Esto tendrás que hacerlo hasta que el tamaño sea un número menor que 0, en cuyo caso ya no tiene sentido seguir dibujando**

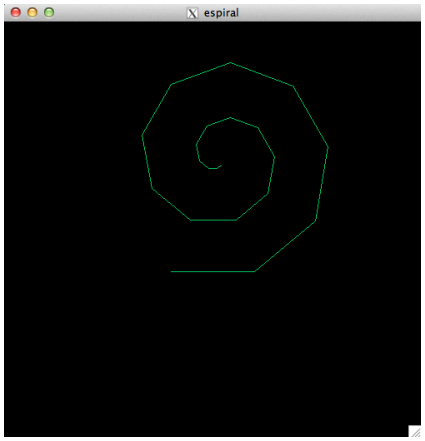
Inicialmente permite al usuario que pueda introducir por teclado el tamaño inicial de la espiral y realiza un giro de 90 grados en cada movimiento. Una vez hecho y probado el programa puedes modificarlo para que también se puedan introducir los grados que se tuerce en cada vuelta.

Ejemplos de ejecución:

```
pc-rosana:pruebasC rosana$ ./ej
Lado inicial:200
Ángulo de giro:90
```

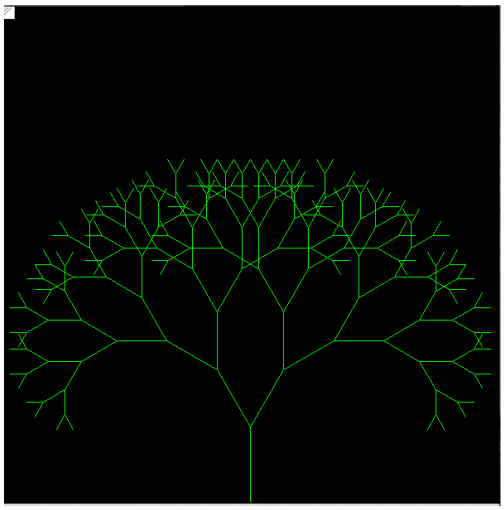



```
pc-rosana:pruebasC rosana$ ./ej
Lado inicial:100
Ángulo de giro:40
```



12. Ejercicio propuesto 7. Resuelve

▶ **Haciendo uso de los gráficos de tortuga, haz un programa que dibuje recursivamente el "árbol" que aparece en la siguiente figura:**



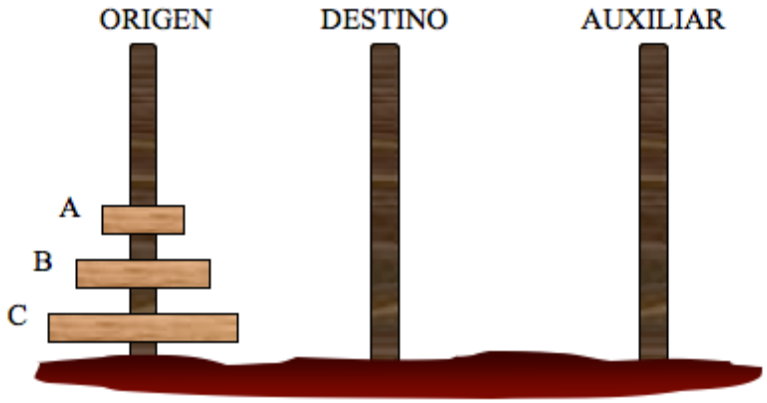
Fíjate en que cada nivel del árbol está formado por una rama que se divide en dos, y lo que tenemos en cada división es una especie de versión más pequeña del árbol completo. Por lo tanto, tendrás que:

- **Dibujar la rama (una línea recta de determinada longitud)**
- **Girar la tortuga sobre sí misma un determinado número de grados a la izquierda, por ejemplo 30 para disponerte a dibujar el "subárbol" de la izquierda.**
- **Dibujar un árbol un poco más pequeño que el tamaño actual**
- **Girar la tortuga a la derecha para disponerte a dibujar el "subárbol" de la derecha (serán -60, ya que tienes que deshacer los 30 que giraste y girar otros 30 más)**
- **Dibujar un árbol un poco más pequeño que el tamaño actual**
- **Volver a girar a la izquierda 30 para quedarte mirando a la horizontal**
- **Volver hacia atrás al punto de partida (en línea recta pero una cantidad negativa, usa la orden "move" en lugar de "draw" ya que no necesitas dibujar)**

Prueba a variar la longitud de la rama o el número de grados, o a hacer árboles asimétricos en los que el número de grados que se gira para el subárbol izquierdo sea distinto al derecho. Mejor aún, modifica el programa para que el usuario pueda introducir los parámetros del árbol (longitud, ángulo izquierdo y ángulo derecho) por teclado.

13. Ejercicio resuelto 5. Repasa

- **En este ejercicio debes resolver el juego de las Torres de Hanoi. Debes realizar un algoritmo recursivo que resuelva este juego. El juego de las Torres de Hanoi empieza del siguiente modo:**



Tenemos una serie de discos ordenados por tamaño en el poste Origen. El objetivo del curso es pasar todos los discos al poste Destino, de manera que en ningún momento quede un disco colocado sobre otro disco de tamaño inferior.

Nos podemos ayudar del poste Auxiliar para colocar discos temporalmente.

6.1 Solución

```
1    #include <iostream>
2    using namespace std;
3
4    void transferir(int,char,char,char);
5
6    int main(void) {
```

```
7       int n;
8       cout << "Bienvenido a las torres de Hanoi"<<endl;
9       cout << "¿Con cuántos discos quieres jugar?";
10      cin >> n;
11      transferir(n,'I','C','D');
12  }
13
14  void transferir(int discos, char desde, char hacia, char
    temp){
15      if (discos>0) {
16          transferir(discos-1,desde,temp,hacia);
17          cout << "Mover disco "<<discos<<" desde "<<desde<<"
    hasta "<<hacia<<endl;
18          transferir(discos-1,temp,hacia,desde);
19      }
20  }
```

10. Errores de programación comunes

- **Definir una función recursiva que no tiene rama de salida (caso base).** Toda función recursiva debe tener una rama de salida que permita poner fin a la recursividad. **En caso contrario, el programa no acabará nunca.**
- **No modificar ningún parámetro en la llamada recursiva.** Alguno de los parámetros tiene que ir cambiando en las sucesivas llamadas recursivas de forma que se tienda a que se cumpla el caso base

ANEXO. Ejemplos de código en Java

Código del ejercicio resuelto 1

Diseña un algoritmo recursivo que muestre por pantalla el inverso de un número.

```
import java.util.Scanner;
```

```
public class Inverso{
```

```
 public static void main(String args[]) {
  int num;
  Scanner sc = new Scanner(System.in);

  System.out.print("Introduzca un número: ");
  num = sc.nextInt();;
  System.out.println();

  System.out.println("El número inverso es: ");
  Inverso(num);
  System.out.println();
 }

 public static void Inverso(int n) {
    int aux;

    if(n > 0) {
      aux = n%10;
      System.out.print(aux);
      Inverso(n/10);
    }
 }
}
```

Código del ejercicio resuelto 2

Escribe un programa que calcule de manera recursiva el factorial de un número.

```
import java.util.Scanner;

public class Funcion{

 public static void main(String args[]) {
  int num;
  int resultado;
  Scanner sc = new Scanner(System.in);
```

```
   System.out.println("Introduzca un número:");
   num = sc.nextInt();

   resultado = funcion(num);
   System.out.println("El resultado es: "+resultado);
  }

  public static int funcion (int n) {
   int result;

   if(n==0)
     result = 1;
   else
     result = n * funcion(n-1);

   return(result);
  }

}
```

Código del ejercicio resuelto 3

En este ejercicio debes resolver el juego de las Torres de Hanoi. Debes realizar un algoritmo recursivo que resuelva este juego.

```
import java.util.Scanner;

public class Hanoi{

 public static void main(String args[]) {
    int n;
    Scanner sc = new Scanner(System.in);

    System.out.println("Bienvenido a las torres de Hanoi");
    System.out.println("¿Cuántos discos?");
    n = sc.nextInt();
    transferir(n,'I','D','C');
```

```
 }

 public static void transferir(int discos, char desde, char
hacia, char temp) {
   if (discos>0){
    transferir(discos-1,desde,temp,hacia);
    System.out.println("Mover disco "+discos+" desde "+desde+"
hasta "+hacia);
    transferir(discos-1,temp,hacia,desde);
   }
 }
}
```