UNIVERSIDAD DE MÁLAGA
E.T.S.I. TELECOMUNICACIÓN

Recursividad

Laboratorio de Programación 2
(1º de Ingeniería de Telecomunicación)

Recursion Simple

1. Escribir una función recursiva para calcular el factorial de un número natural n. Emplear el tipo `long int`. Comprobar su funcionamiento calculando los factoriales de 0, 1, 2, 3, 4 y 5. ¿Qué ocurre al tratar de calcular el factorial de 13? ¿Podría calcularse el factorial de 13 con una versión iterativa? ¿Cómo hay que modificar la función para poder calcular el factorial de números naturales mayores o iguales a 13?
2. Escribir una función recursiva EscribeBlancos(n) que imprima n caracteres blancos consecutivos.
3. Escribir una función recursiva `Invierte` que acepte caracteres por teclado hasta recibir un retorno de carro y los imprima en orden inverso al de lectura. Los caracteres se leerán uno a uno y no deben almacenarse en un array; basta emplear una sola variable local a la función `Invierte` de tipo `char`.
4. Modificar la función factorial recursiva del ejercicio 1 para que al ejecutarse imprima una traza de su ejecución como la siguiente:

```
entrando en Factorial(3)
   entrando en Factorial(2)
      entrando en Factorial(1)
         entrando en Factorial(0)
         saliendo de Factorial(0)
      saliendo de Factorial(1)
   saliendo de Factorial(2)
saliendo de Factorial(3)
```

   Sugerencia: añadir a la función factorial un segundo parámetro que indique el número de espacios en blanco que deben aparecer en el margen izquierdo de sus mensajes de traza y emplear la función `EscribeBlancos(n)` del ejercicio 2 para imprimir este margen.
5. Escribir una función recursiva que calcule el n-ésimo número de Fibonacci.
6. Modificar la función del apartado anterior para que muestre una traza de su ejecución como la siguiente:

```
entrando en Fibonacci(4)
  entrando en Fibonacci(3)
    entrando en Fibonacci(2)
      entrando en Fibonacci(1)
      saliendo de Fibonacci(1)
      entrando en Fibonacci(0)
      saliendo de Fibonacci(0)
    saliendo de Fibonacci(2)
    entrando en Fibonacci(1)
    saliendo de Fibonacci(1)
  saliendo de Fibonacci(3)
  entrando en Fibonacci(2)
    entrando en Fibonacci(1)
    saliendo de Fibonacci(1)
    entrando en Fibonacci(0)
    saliendo de Fibonacci(0)
  saliendo de Fibonacci(2)
saliendo de Fibonacci(4)
```

7. Escribir una función recursiva Pascal(i,j) que calcule el elemento i, j del triángulo de Pascal, que sigue el siguiente patrón:

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5   10  10  5   1
. . . . . . . . . . . . . .
```

Como puede apreciarse, los elementos en el borde del triángulo son 1's, y el resto de los elementos son iguales a la suma de los dos elementos que hay sobre ellos.

8. Escribir una función recursiva `Palindromo(c,i,j)` que determine si la subcadena contenida entre las posiciones i y j de la cadena c es un palíndromo.

Recursion mutua

9. Escribir dos funciones mutuamente recursivas para calcular suma de los n primeros términos de la serie alternada: $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} \cdots - \frac{1}{2i} + \frac{1}{2i+1} \cdots$
Sugerencia: Escribir una función `Pares(i)` que calcule la suma hasta el término i (i par) y otra función `Impar(i)` que calcule la suma hasta el término i (i impar). Estas funciones deben llamarse una a la otra, hasta que se haya completado la suma de los n primeros términos. Para sumar los n primeros términos, escribir una función `SumaSerie(n)`, que llamará a `Par(n)` o `Impar(n)` dependiendo de la paridad de n.

Recursión no primitiva

10. La función de Ackerman se define para dos valores naturales dados m y n como sigue:
$$\begin{aligned} Ackerman(0,n) &= n+1 \\ Ackerman(m,0) &= Ackerman(m-1,1) \\ Ackerman(m,n) &= Ackerman(m-1, Ackerman(m,n-1)) \end{aligned}$$
Se trata de un ejemplo muy conocido de función recursiva no primitiva (la función aparece como argumento de sí misma). Escribir una función recursiva para calcular la función de Ackerman. Comprobar su funcionamiento calculando los siguientes valores: `Ackerman(1,1)= 3`, `Ackerman(2,1)= 5`, `Ackerman(3,3)= 61`, `Ackerman(3,5)= 253`. ¿Qué ocurre al intentar calcular `Ackerman(5,1)`?

11. Desarrollar un algoritmo recursivo cuya función sea devolver el número de `'#'` unidos en una malla. Un `'#'` está unido a otro `'#'` si son vecinos horizontal, vertical o diagonalmente.

		#	#				#
	#		#	#		#	
						#	
	#		#	#		#	#
#					#		

12. Escribe un algoritmo recursivo que ordene un array de la siguiente forma:
    - Sea `k` el índice del elemento mitad del array.
    - Ordena los elementos hasta `a[k]`, incluyéndolo.
    - Ordena los elementos siguientes.
    - Mezcla los dos subarrays en un único array ordenado.

    Este método de clasificación interna se denomina *ordenación por mezcla*, o *Mergesort*.