Ejercicios sobre recursividad

11 de febrero de 2003

1. Implementa una función recursiva que devuelva la suma de los dígitos de un número natural, que se le pasa por parámetro.

2. Implementa una función recursiva que imprima por pantalla los valores desde 1 hasta un número introducido por el usuario.

3. Implementa una función recursiva que imprima por pantalla los valores desde un número introducido por el usuario hasta 1.

4. Implementa una función recursiva que devuelva el número de dígitos de un número natural.

5. Implementa una función recursiva que calcule el producto de dos números naturales (se supone que no está disponible el operador `*`).

6. Implementa una función recursiva que, dado un número binario (representado mediante un entero), devuelva su valor en base decimal (*p.e.* `101 = 5`, `11111 = 31`).

7. Implementa una función recursiva que devuelva el resultado de la siguiente expresión, para un número natural x pasado por parámetro:

$$f(x) = x^2 + (x-1)^2 + (x-2)^2 + \ldots + 2^2 + 1^2$$

8. Implementa una función recursiva que lea desde teclado hasta encontrar un carácter de salto de línea (`\n`), y muestre por pantalla lo que ha leído, pero al revés (no utilizar vectores ni bucles).

9. Implementa una función recursiva que imprima por pantalla un número natural (se supone que sólo está disponible la función `putchar`).

10. Implementa una función recursiva que calcule el logaritmo entero de un número a en una base b, ambos introducidos por el usuario. *Ayuda:* ten en cuenta que $\log_b(a/b) = \log_b a - log_b b$.

11. Implementa una función recursiva que, dado un número natural, devuelva otro número que tenga los dígitos del primero, pero al revés. *Ayuda:* utiliza un parámetro auxiliar, donde vayas acumulando los resultados parciales.

12. Implementa una función que, dado un vector de `TAM` enteros, devuelva el mínimo y el máximo. *Ayuda:* utiliza un parámetro auxiliar que indique la porción del vector que has recorrido, y para que la función devuelva dos enteros, define una estructura.

13. En el gran templo de Benarés, bajo la cúpula que marca el centro del mundo, se encuentra una bandeja de cobre en la que hay fijadas tres agujas de diamante, cada una de un codo de alto y del grosor del cuerpo de una abeja. En una de esas agujas, en la creación, Dios puso sesenta y cuatro discos de oro puro, estando el mayor sobre la bandeja y los demás ordenados de mayor a menor sobre el primero. Esta es la Torre de Brahma. Día y noche sin parar los monjes mueven los discos entre las agujas siguiendo las leyes inmutables de Brahma, que establecen que el monje en activo no debe mover más de un disco a la vez y que no puede poner un disco en una aguja que contenga un disco menor. Cuando los sesenta y cuatro discos se hayan movido desde la aguja original a una de las otras dos, la torre, el templo y los monjes se convertirán en cenizas, y con un trueno el mundo desaparacerá.

    Con el tiempo, la Torre de Brahma se ha convertido en la Torre de Hanoi.

    a) Implementa un algoritmo recursivo para solucionar el problema de la Torre de Hanoi, para un número de discos especificado por parámetro.

    b) Calcula el número de movimientos necesarios para resolver el problema con `n` discos.

    c) Suponiendo que los monjes mueven un disco por segundo, ¿cuánto tiempo nos queda de vida? (según las últimas noticias, el universo tiene entre 13 y 14 mil millones de años).

    Nota: En `Emacs`, pulsa `Alt-x` y escribe `hanoi`. Prueba también a pulsar `Esc`, luego `:`, y escribe `(hanoi 5)`.

14. Implementa una función recursiva para mostrar por pantalla todas las permutaciones de los caracteres de una cadena que se le pasa por parámetro.

15. La mayoría de los teléfonos móviles incorporan en el teclado numérico (excepto en las teclas 0 y 1) una serie de letras, que permiten, por ejemplo, componer mensajes de texto o recordar números de teléfono

mediante la palabra que lo compone (por ejemplo es más fácil recordar "miércoles" que "643726537"):

1	ABC 2	DEF 3
GHI 4	JKL 5	MNO 6
PQRS 7	TUV 8	WXYZ 9
	0	

Implementa una función recursiva que, dado un número, escriba por pantalla todas las posibles palabras que se corresponden con dicho número (utiza números pequeños).

16. Transforma las soluciones recursivas de los ejercicios 3 y 11 a iterativas, utilizando la transformación recursivo-iterativa de funciones lineales finales.

17. Implementa la versión iterativa de los ejercicios 4, 5, 6, 8 y 9. Compara ambas soluciones (la versión recursiva suele ser más compacta y sencilla de entender).

18. Implementa una función recursiva para calcular un elemento del triángulo de Pascal, que tiene la siguiente forma:

```
                    1
                  1   1
                1   2   1
              1   3   3   1
            1   4   6   4   1
          1   5  10  10   5   1
        1   6  15  20  15   6   1
      1   7  21  35  35  21   7   1
    1   8  28  56  70  56  28   8   1
```

Cada elemento de dicho triángulo es la suma de los dos números que tiene por encima. Los valores extremos de cada fila valen 1.

a) Implementa una función recursiva con la siguiente cabecera:

```
int pascal(int nfila, int nelem)
```

donde, `nfila` es la fila del triángulo, y `nelem` la posición del elemento que se desea calcular dentro de la fila.

b) Implementa la función `main` correspondiente para construir por pantalla el triángulo de Pascal utilizando la función anterior.

19. Implementa una función recursiva para decidir si una palabra es un palíndromo o no.

20. Implementa una función recursiva que, dado un número entero, muestre por pantalla su valor en binario.

21. Implementa una función recursiva que calcule el número de llamadas que se generan para calcular el número de Fibonacci mediante el algoritmo recursivo básico, para un cierto número `N`, pasado por parámetro.

22. Dadas las siguientes funciones:

```
int g(int n, int a, int b) {
  if (n==0) return a;
  else return g(n-1,b,a+b);
}

int f(int n) {
  return g(n,0,1);
}
```

Calcula los valores de `f(0)`, `f(1)`, `f(2)`...`f(5)`. ¿Qué función matemática está calculando `f`? Indica cuántas sumas se realizan en función de `n`. Indica la complejidad del algoritmo.

Ten en cuenta que todo el trabajo lo realiza la función recursiva `g`, y que `f` únicamente se utiliza para establecer los valores iniciales correctos de los parámetros `a` y `b`. Esta técnica se utiliza a menudo cuando se trabaja con recursión.

23. Calcula la complejidad computacional de las soluciones de los ejercicios: 1, 5, 10, 12, 13 y 14.

24. Resuelve las siguientes relaciones de recurrencia por el método del desplegado:

a)

$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ T(n-1) + n + 1 & \text{si } n > 1 \end{cases}$$

b)

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ T(n-2) + n & \text{si } n > 1 \end{cases}$$

c)

$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ T(n/2) + n + 1 & \text{si } n > 1 \end{cases}$$

d)

$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ 2T(n/2) + n + 1 & \text{si } n > 1 \end{cases}$$

e)

$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ 4T(n/2) + 1 & \text{si } n > 1 \end{cases}$$

f)

$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ 4T(n/2) + n + 1 & \text{si } n > 1 \end{cases}$$

g)

$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ 3T(n/2) + n + 1 & \text{si } n > 1 \end{cases}$$

25. Demuestra por inducción que la suma de los N primeros números impares es N^2:

$$1 + 3 + 5 + \ldots + (2N - 1) = N^2$$

26. Demuestra por inducción que la suma de los primeros N enteros pares es $N^2 + N$.

27. Demuestra por inducción que las siguientes fórmulas son correctas:

   a) $1 + 2 + 4 + 8 + \ldots + 2^{N-1} = 2^N - 1$

   b) $1 + 3 + 9 + 27 + \ldots + 3^N = \frac{3^{N+1}-1}{2}$

   c) $1 \times 1 + 2 \times 2 + 3 \times 4 + 4 \times 8 + \ldots + N \times 2^{N-1} = (N - 1)2^N + 1$

   d) $\forall n \in \mathbb{N} \quad 2^n < n!$, si $n \geq 4$

28. Demuestra por inducción que $10^n - 1$ es divisible por 9 $\forall n \in \mathbb{N}$.

29. Demuestra por inducción que las soluciones obtenidas en el ejercicio 24 son correctas.

30. Imagina una pila de bolas de cañón. En el vértice superior hay una, que se apoya en otras cuatro. Estas cuatro se apoyan a su vez en una capa de nueve, etc. Utilizando inducción matemática, demuestra que el número de bolas en función del número de capas es:

$$\frac{N(N+1)(2N+1)}{6}$$

31. Partiendo del resultado obtenido en el ejercicio 25, diseña una recurrencia para calcular el cuadrado de cualquier número a partir del cuadrado del anterior. Implementa entonces la siguiente función recursiva, que devuelve el cuadrado de `n`:

    ```
    int cuadrado(int n)
    ```

32. Cambio de moneda. Diséñese un algoritmo recursivo que dada una cierta cantidad M, efectúe el cambio a monedas de 2, 1, 0.5, 0.2, 0.1, 0.05, 0.02 y 0.01 euros, devolviendo el número de monedas de cada tipo que constituyen el cambio (eventualmente cero para alguno de los tipos). Se desea que el número de monedas obtenido sea el mínimo. Los únicos operadores disponibles serán los de suma y resta.

33. Diseñar un algoritmo recursivo que, dado un natural x y un vector `v` con n naturales, determine si x es igual a la suma de todas las componentes del vector. El perfil de la función debe ser el siguiente, y no se pueden utilizar parámetros auxiliares.

    ```
    int es_suma_comp(int vector[], int tam_vector, int x);
    ```

34. Dado un vector de n números enteros, se desea diseñar un algoritmo recursivo que dado el vector devuelva:

    - la posición del último número par que aparece en el vector, o
    - el valor 0 si ningún elemento del vector es par.

35. Diseñar una función que determine si una matriz `M` de dimensión $n \times n$, con $n \geq 1$, es o no simétrica con respecto a la diagonal principal. Para ello, diseñar una función recursiva auxiliar que compruebe si la fila `M[i][0..i-1]` es simétrica a la columna `M[0..i-1][i]`.

36. Diseñar una función recursiva que, con un coste lineal, busque en un vector de enteros un elemento que coincida con la suma de todos los elementos que tiene a su izquierda.

37. Diseñar una función recursiva para calcular multiplicaciones según el siguiente método conocido como del *campesino egipcio*:

$$mult(x,y) = \begin{cases} 0 & \text{si } y = 0 \\ x & \text{si } y = 1 \\ mult(2 * x, y/2) & \text{si } y \geq 2, \text{ e } y \text{ es par} \\ mult(2 * x, \lfloor y/2 \rfloor) + x & \text{si } y \geq 2, \text{ e } y \text{ es impar} \end{cases}$$

    Demostrar su corrección y calcular su coste.

38. Demuestra que el *i-ésimo* número de Fibonacci es menor que $\left(\frac{5}{3}\right)^i$, para cualquier natural mayor que cero.