Metodología de la Programación
Grado en Ingeniería Informática
Guión de Prácticas Nº 1

RECURSIVIDAD

Objetivos

- Aprender a resolver problemas de forma recursiva.
- Dominar los métodos de transformación de algoritmos recursivos.

En las prácticas de este tema, el alumno deberá:

- Implementar una solución recursiva para todos los problemas del bloque I que se presentan a continuación.
- Obtener para todos los problemas del bloque II una función recursiva final equivalente a la función recursiva que se presenta y las correspondientes soluciones iterativas, detallando todos los pasos en cada una de las transformaciones, siguiendo los métodos de transformación explicados en la teoría de la asignatura.
- Implementar en C las versiones recursivas e iterativas del paso anterior.

BLOQUE I - Implementación

1.- Escribe una función recursiva que calcule el Máximo Común Divisor de dos números *a* y *b* utilizando el algoritmo de Euclides.

2.- Escribe una función recursiva *cifras* que reciba un número en base 10 y devuelva cuántas cifras tiene ese número.

3.- Dado un vector A de n enteros ordenado crecientemente, diseñe una función recursiva que devuelva su moda, es decir, el valor más frecuente de A.

4.- Diseñe una función recursiva que determine si en un vector A de n enteros existen dos parejas consecutivas de elementos tales que sus sumas sean idénticas.

5.- Si *conj(n,k)* representa la cantidad de diferentes conjuntos de *k* personas que pueden formarse, dadas *n* personas entre las cuales elegir. Por ejemplo, *conj*(4,3) = 4, porque dadas cuatro personas A, B, C y D hay cuatro conjuntos posibles de tres personas: ABC, ABD, ACD y BCD. En general se cumple la siguiente relación:
conj(n,k) = conj(n-1,k) + conj(n-1,k-1)

Escribe una función recursiva para calcular *conj(n,k)* para $n,k \geq 1$.

6.- Dado un vector ordenado crecientemente A[1..n], n ≥ 1, diseña un algoritmo que calcule de forma recursiva la longitud de la escalera más larga, es decir, la longitud de la secuencia más larga de valores consecutivos que se encuentre en A.

7.- Dado un vector *n* de enteros, se dice que un elemento del vector es elemento mayoritario si este entero aparece estrictamente más de *n*/2 veces en dicho vector. Diseña un algoritmo que determine de forma recursiva si el vector A[1..n] contiene un elemento mayoritario y la primera posición que ocupa.

8.- Dado un vector A de n enteros y un número natural k, 1 ≤ k ≤ n, diseñe un procedimiento recursivo que intercambie los k primeros elementos de A con los elementos de las k últimas posiciones, sin hacer uso de un vector auxiliar.

9.- Diseña un algoritmo que calcule de forma recursiva la suma de todos los elementos *i* de un vector *a* de *n* enteros que cumplen la siguiente propiedad:

$$\boxed{1 \le i \le \left\lceil \frac{n}{2} \right\rceil - 1 : a[i] > a[2*i] \wedge a[i] > a[2*i+1]}.$$

10.- Diseña un algoritmo que determine de forma recursiva si en un vector A de *n* enteros se cumple: $\boxed{\forall \alpha : 1 \le \alpha \le \left\lceil \frac{\boldsymbol{n}}{2} \right\rceil : \boldsymbol{A}[\alpha] = \boldsymbol{A}[\boldsymbol{n} - \alpha + 1]}$.

11.- En un vector de enteros se genera un «cambio de tendencia» cuando dada una secuencia creciente o decreciente de números que ocupan posiciones consecutivas del vector, el elemento que le sucede es inferior o superior respectivamente. Dado un vector de N enteros, diseñe una función recursiva que calcule el número de «cambios de tendencia» que contiene dicho vector.

12.- Dado un vector de enteros estrictamente positivos A[1..n], n ≥ 1, diseña un algoritmo que obtenga recursivamente el número de parejas (j,k), 1≤ j ≤ n, 1≤ k ≤ n tales que $\boxed{\sum_{i=1}^{j} A[i] = \sum_{i=k}^{n} A[i]}$.

BLOQUE II – Transformación

Se supone la existencia del tipo *Vect* definido como: **vector**[N] de entero: Vect

13.- entero **función** fun(E Vect: x, E entero: n E entero: i)
{ x = A[1..n] ∧ n≥0 ∧ 0≤ i≤n }
inicio
  si i=0 **entonces**
    devolver 0
  si_no
    devolver x[i] + fun(x, n, i-1)
  fin_si

{devuelve la suma de los *i* primeros elementos del vector}
fin_función

14.- entero **función** fun(E Vect: x, E entero: n, E entero: i)
{ x = A[1..n] ∧ n≥0 ∧ 0≤ i≤n }
inicio
  si i=0 **entonces**
    devolver 0
  si_no
    devolver x[i]*x[i] + fun(x, n, i-1)
  fin_si
{devuelve $\sum_{\alpha=1}^{i} x[\alpha]^2$ }
fin_función

15.- entero **funcion** fun (E Vect: x, E Vect: y, E Vect: z, E entero: n, E entero: i)
{x = A[1..n] ∧ y = B[1..n] ∧ z=C[1..n] ∧ 1≤ i≤n }
inicio
  si i=n **entonces**
    devolver x[i]*y[i] + y[i]*z[i]
  si_no
    devolver x[i]*y[i] + y[i]*z[i] + 5*fun(x, y , z, n, i+1)
  fin_si
{devuelve $\sum_{\alpha=i}^{n} (x[\alpha] * y[\alpha] + y[\alpha] * z[\alpha]) * 5^{\alpha-i}$ }
fin_función

16.- real **funcion** fun (E Vect: x, E Vect: y, E entero: n, E entero: i)
{x = A[1..n] ∧ y = B[1..n] ∧ i≤n ∧ i≥1 ∧ n>0 }
inicio
  si i=n **entonces**
    devolver x[i]*y[n-i+1]
  si_no
    devolver x[i]*y[n-i+1] + (i+1)*fun(x, y, n, i+1)
  fin_si
{devuelve $\sum_{\alpha=i}^{n} \frac{\alpha!}{i!} (x[\alpha] * y[n-\alpha+1])$ }
fin_función

17.- entero **funcion** fun (E Vect: x, E Vect: y, E entero: n, E entero: i)
{x = A[1..n] ∧ y = B[1..n] ∧ 1≤i ∧ i≤n}
inicio
  si i>n **entonces**
    devolver 1
  si_no
    devolver (6*x[i] + 6*y[i]) * fun(x, y, n, i+1)
  fin_si

{devuelve $\prod_{\alpha=i}^{n} 6*(x[\alpha]+y[\alpha])$ }

fin_función

18.- real **funcion** fun (E Vect: x, E Vect: y, E entero: n, E entero: i)

{x = A[1..n] ∧ y = B[1..n] ∧ 1≤ i≤n }

inicio

  si i=n **entonces**

    devolver 3*x[i]*y[i]

  si_no

    devolver 3*x[i]*y[i] + $\frac{1}{i+1}$*fun(x, y, n, i+1)

  fin_si

{devuelve $\sum_{\alpha=i}^{n} \frac{i!}{\alpha!}(3*x[\alpha]*y[\alpha])$ }

fin_función

19.- real **funcion** fun (E Vect: x, E Vect: y, E entero: n, E entero: i)

{x = A[1..n] ∧ y = B[1..n] ∧ i≤n ∧ i≥1 ∧ n>0 }

inicio

  si i=n **entonces**

    devolver i*x[i]

  si_no

    devolver i* x[i] + (n-i)*fun(x, y, n, i+1)

  fin_si

{devuelve $\sum_{\alpha=i}^{n} \frac{\alpha!}{i!}(x[\alpha]*\alpha)$ }

fin_funcion

20.- entero **funcion** fun (E Vect: x, E Vect: y, E entero: n, E entero: i)

{x = A[1..n] ∧ y = B[1..n] ∧ i≤n ∧ i≥1 ∧ n>0 }

inicio

  si i=n+1 **entonces**

    devolver 0

  si_no

    devolver x[i]*y[n-i+1] + 3*fun(x, y, n, i+1)

  fin_si

{devuelve $\sum_{\alpha=i}^{n} 3^{\alpha-i}(\boldsymbol{x}[\alpha]*\boldsymbol{y}[\boldsymbol{n}-\alpha+1])$ }

fin_funcion