Metodología de la Programación
Grado en Ingeniería Informática
Guión de Prácticas

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:

- Diseñar un subalgoritmo recursivo para todos los problemas del bloque I que se presentan a continuación.
- Implementar en C los subalgoritmos recursivos del bloque I.

- 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 de subalgoritmos recursivos

1.- 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.

2.- Diseña una función recursiva que calcule la *i*-ésima cifra de un entero *n*.

3.- Diseña un algoritmo que localice de forma recursiva, en la misma pasada, el máximo y el mínimo de un vector dado no vacío.

4.- Diseña un algoritmo que calcule de forma recursiva el producto escalar de dos vectores de *n* elementos enteros, $n \geq 0$.

5.- 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.

6.- Diseña 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.

7.- 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:

$$1 \le i \le \left(\frac{n}{2}\right) - 1 : A[i] > A[2*i] \wedge A[i] > A[2*i+1]$$

.

8.- 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.

9.- 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 $\sum_{i=1}^{j} A[i] = \sum_{i=k}^{n} A[i]$.

BLOQUE II – Transformación de subalgoritmos recursivos

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

10.-
```
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
```

11.-
```
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  Σ_{α=i}^{n} α!/i! (x[α] * y[n-α+1]) }
fin_función
```

12.- 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

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