Estrategias de Programación y Estructuras de Datos

Tema 2 – Recursividad. Ejercicios Propuestos

A continuación, se proponen ejercicios para complementar el estudio del tema 2 de la asignatura de Estrategias de Programación y Estructuras de Datos. Los ejercicios 5 y 6 utilizan estructuras de datos que se estudian a fondo en temas posteriores; recomendamos que se vuelva sobre ellos una vez que se hayan estudiado las listas y los árboles.

1. Función factorial

El factorial de un número natural n se define como

- 1 , si n=0
- n * factorial(n-1), en caso contrario.

Identifíquense las partes de esta definición recursiva:

- cuál sería el caso no recursivo, y qué haría el programa en este caso.
- Cuál sería el caso recursivo, cuál sería el argumento de la llamada recursiva, y cómo se obtiene el resultado final a partir del resultado de la llamada recursiva.

2. Serie de Fibonacci

El elemento n-ésimo de la serie de Fibonacci se define como

- 1, si n=0 o n=1
- Fibonacci(n-1) + Fibonacci(n-2) si n>1

Identifíquense las partes de esta definición recursiva como en el ejercicio anterior. ¿Por qué no sería eficiente una implementación directa de la serie de Fibonacci según esa definición? ¿Qué alternativas se le ocurren?

3. Exponencial de un número

Considérese la siguiente definición recursiva de la exponenciación de enteros a^b:

$a^b = 1$, si b=0
$= a^{b/2} * a^{b/2}$, si b>0

Identifíquense las partes de esta definición recursiva como en los ejercicios anteriores. ¿Es correcta? ¿Cómo se convertiría en un programa recursivo? ¿Sería eficiente? ¿Qué problemas habría que solucionar?

4 Más funciones numéricas recursivas

Explica qué calculan, y qué problemas tienen en su caso, cada una de las siguientes definiciones recursivas de funciones sobre enteros:

$f_1(n) = 2$ si $n=1$
$\quad 2+f_1(n-1)$ si $n>1$

$f_2(n) = 1$ si $n=0$
$\quad e*f_2(n-1)$ si $n>0$

$f_3(n) = 1$ si $n=3$
$\quad = 2*f_3(n-2)$ si $n\neq 3$

$f_4(n) = 0$ si $n=0$
$\quad = 3*f_4(n-1)$ si $n>0$

$f_5(n) = f_5(n-1)$ si $n > 5$
$\quad = 3*f_5(n-2)$ si $n < 5$

5. Lista módulo

La entrada del programa es una lista de números enteros, y la salida es el resultado de sumarlos todos.

Utilícese la definición recursiva de una lista, es decir:

Una lista de enteros es, o bien una lista vacía, o bien un primer elemento y una lista con el resto de elementos. Supónganse disponibles los métodos isEmpty? (devuelve cierto si la lista está vacía y falso en caso contrario), first (devuelve el primer elemento de la lista) y rest (devuelve una lista con todos los elementos menos el primero).

6. Árbol de suma nula

Diseñar un programa cuya entrada es un árbol binario de enteros, y cuya salida es cierto si la suma de todos los enteros del árbol es cero, y falso en caso contrario.

Para diseñar el programa, utilícese la siguiente definición recursiva de un árbol binario de enteros:

Un árbol binario es, o bien un árbol vacío, o bien un nodo raíz (un valor entero) y dos árboles (que llamaremos árbol izquierdo y árbol derecho).

Supónganse disponibles los métodos *isEmpty?* (devuelve cierto si el árbol es vacío y falso en caso contrario), *root* (devuelve el valor del nodo raíz), *leftChild* (devuelve el subárbol izquierdo) y *rightChild* (devuelve el subárbol derecho).