Ejercicios de seudocódigo Arboles Binarios de Búsqueda

EJERCICIO 1

CONTEXTO

Los árboles binarios de búsqueda son muy utilizados para almacenamiento de información y búsquedas sobre la misma.
Supongamos que, en una estructura de este tipo, se han almacenado los datos de todos los alumnos de una Facultad. Cada alumno está representado mediante un identificador único (su CI, numérico, sin código de verificación), su nombre, apellido y carrera en la que está matriculado (estos tres son alfanuméricos).
Una de las tareas típicas que se suele realizar es consultar por los alumnos de una cierta carrera. Para evitar realizar una búsqueda extensa sobre todos los alumnos de la facultad, se decide generar otras estructuras que permitan hacerlo en forma más eficiente.
Se desea entonces, dado un árbol binario de búsqueda (TDA TABB) que contenga todos los alumnos de la facultad (con los datos indicados más arriba, habiendo sido insertados de acuerdo a su número de CI), y dado el nombre de una cierta carrera, crear un nuevo árbol que indexe la información de los alumnos correspondientes, por Apellido (se desea poder imprimir la lista de alumnos de la carrera ordenada por Apellido en la forma más eficiente).

Tipo Alumno
CI : numérico
Nombre: alfanumérico
Apellido: alfanumérico
Carrrera: alfanumérico

De TABB armarIndiceCarrrera (de tipo alfanumérico unaCarrera) devuelve TABB

NOTAS IMPORTANTE:

- Se deben desarrollar los métodos de TArbolBB y de TElementoAB
- Se debe evitar duplicar información

Ejemplo de invocación:

TABB facultadDeIngeniería // contiene todos los alumnos de ingeniería, por CI
TABB ingenieriaInformatica = facultadDeIngeniería.armarIndiceCarrera(“Ingeniería Informática”)

CONSIDERACIONES IMPORTANTES

- Se deben respetar las normas publicadas sobre la escritura de seudocódigo (lenguaje natural, pre y post condiciones, seudocódigo detallado sobre esta implementación del TDA.
- Desarrollar en lenguaje natural, a partir de las post-condiciones indicadas, los casos de prueba que se estimen convenientes.
- Se deben desarrollar completamente todas las operaciones invocadas.

EJERCICIO 2

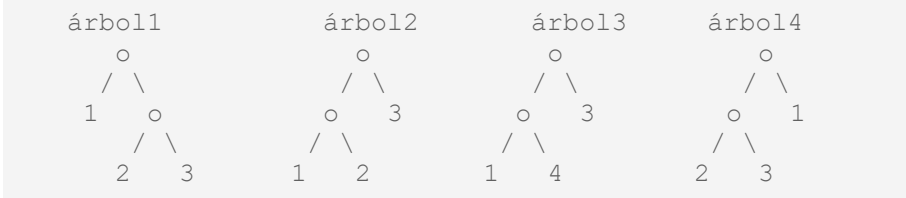
Los árboles binarios se pueden utilizar para innumerables propósitos: representar expresiones aritméticas, implementar árboles de decisión (donde las variables tengan dos clases), y muchas otras.
A menudo resulta necesario determinar si un cierto árbol es idéntico a otro, es decir, si tiene la misma estructura y los nodos correspondientes contienen la misma información.

Se desea:
Diseñar un algoritmo que permita saber si dos árboles binarios son idénticos.

Firma (a nivel del árbol): de TDA Arbol Binario **esIdentico (Arbol Binario otroArbol) devuelve boolean**

EJERCICIO 3

Dos árboles binarios tienen iguales los bordes si tienen exactamente las mismas hojas leídas de izquierda a derecha, independientemente de nodos interiores. Por ejemplo,



Los bordes de los árboles 1 y 2 son iguales, aunque tiene distintas estructuras internas. El árbol 3 no tiene el mismo borde que el 1 o el 2, debido al nodo 4. El árbol 4 tampoco tiene el mismo borde que el 1 debido al orden en que se leen las hojas.

Se desea:
Diseñar un algoritmo que permita saber si dos árboles binarios tienen iguales bordes.

Firma (a nivel del árbol): de TDA Arbol Binario **igualesBordes (Arbol Binario otroArbol) devuelve boolean**

NOTA: el algoritmo debe utilizar la menor cantidad de memoria auxiliar posible, y tener el mejor orden de tiempo de ejecución posible.

EJERCICIO 4

A menudo se desea conocer, para un cierto conjunto de elementos, cuántos son menores que un argumento dado.
Supongamos que tenemos un conjunto de elementos con claves enteras almacenados en un árbol binario de búsqueda.
Se desea desarrollar un algoritmo para el TDA Arbol Binario de Búsqueda TABB que, dada una clave o argumento de búsqueda, devuelva la cantidad total de elementos del árbol que tienen claves menores.

De TDA TABB menoresQue (unaclave) devuelve entero;

Este algoritmo debe visitar la menor cantidad posible de nodos del árbol.

EJERCICIO 5

A menudo se desea para un cierto conjunto de elementos, imprimir o procesar solamente un subconjunto de los mismos, cuyas claves se encuentren en un cierto rango determinado.
Supongamos que tenemos un conjunto de elementos con claves enteras almacenados en un árbol binario de búsqueda.
Se desea desarrollar un algoritmo para el TDA Arbol Binario de Búsqueda TABB que, dado un rango de valores (extremo inferior, extremo superior) imprima en forma ordenada todas las claves contenidas en el árbol que estén dentro del rango.

De TDA TABB imprimirRango(de tipo entero: inferior, superior);

Este algoritmo debe visitar la menor cantidad posible de nodos del árbol.