EjercicioABB1

Estructuras de Datos

Tema 4: árboles

1º Grado en Ingeniería de la Computación
© Profesor Dr. Carlos Grima Izquierdo (www.carlosgrima.com)
URJC (www.urjc.es)

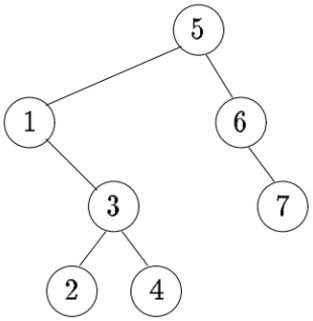
Programa en C++ la clase “ArbolBinarioDeBusqueda” (no se permiten duplicados). Por cada método, habrá que enunciar su O() temporal en el peor caso, con la mejor y peor topología. La clase tendrá al menos las siguientes características:

- Necesitaremos un struct “Nodo” que contenga un int (el elemento) y punteros al padre (NULL si no tiene), hijo izquierdo (NULL si no tiene) e hijo derecho (NULL si no tiene).

- La clase “ArbolBinarioDeBusqueda” tendrá los siguientes componentes (intenta ir paso a paso y probando todo antes de programar lo siguiente):

    - Atributos: Puntero a la raíz del árbol, y número de nodos del árbol ("n").
    - Constructor que construirá un árbol vacío.
    - Un método privado (recursivo) para buscar un elemento dentro del árbol. Tendrá dos parámetros: el puntero al nodo raíz del subárbol en donde buscar (precondición: no debe ser NULL), y el elemento a buscar. Si encuentra el elemento, devolverá el puntero al nodo que lo contiene. Si no encuentra el elemento, devolverá el puntero al nodo que sería su padre en caso de que posteriormente quisiéramos insertar el elemento.
    - Insertar un elemento (método público): buscará la posición en donde deberíamos insertar el nuevo elemento (para ello usará el método privado recursivo descrito antes) y lo meterá ahí (sólo como hoja). El parámetro es el elemento a insertar. La precondición es que no exista dentro del árbol el elemento que queremos insertar.
    - Buscar un elemento (método público). Buscará un elemento. Si lo encuentra, devuelve verdadero. Si no, devuelve falso. Para ello usará el método privado recursivo que hemos descrito antes. Tendrá un único parámetro: el elemento a buscar.
    - Imprimir (método público). Imprimirá el árbol en forma de esquema numerado, al estilo de anteriores ejercicios sobre árboles. Si el árbol está vacío, imprimirá el mensaje "Arbol vacio". La única diferencia con ejercicios anteriores sobre árboles será que, por cada nodo impreso (exceptuando la raíz del árbol completo), entre paréntesis imprimiremos si es hijo izquierdo o hijo derecho de su padre. No tendrá ningún parámetro. Al estilo de anteriores ejercicios sobre árboles, este método público llamará a uno privado recursivo. Este método privado recursivo sí tendrá tres parámetros: la raíz del subárbol a imprimir, el número de tabulaciones con que se imprimirá dicha raíz, y un número entero que nos diga si la raíz del subárbol es hija izquierda de su padre, hija derecha de su padre, o no tiene padre.
    - Destructor que liberará la memoria de todos los nodos. El destructor llamará a un método privado recursivo que destruirá un subárbol, como en la actividad EjercicioGenealógico.

- Pruébalo todo con un main en el cual vayamos insertando hasta conseguir el árbol de la siguiente figura. Cada vez que insertemos, imprimiremos el árbol entero. La inserción debe hacerse mediante un bucle a partir de un array, al igual que en ejercicios anteriores sobre árboles. A continuación, buscaremos todos los elementos insertados, uno por uno (sirviéndonos del mismo array). Primero los buscaremos en el mismo orden de inserción, y a continuación en el orden inverso al de inserción.




```
Consola de depuración de Microsoft Visual Studio
Insercion de 5:
5
Insercion de 1:
5
        1 (izquierdo)
Insercion de 6:
5
        1 (izquierdo)
        6 (derecho)
Insercion de 3:
5
        1 (izquierdo)
                3 (derecho)
        6 (derecho)
Insercion de 7:
5
        1 (izquierdo)
                3 (derecho)
        6 (derecho)
                7 (derecho)
Insercion de 2:
5
        1 (izquierdo)
                3 (derecho)
                        2 (izquierdo)
        6 (derecho)
                7 (derecho)
Insercion de 4:
5
        1 (izquierdo)
                3 (derecho)
                        2 (izquierdo)
                        4 (derecho)
        6 (derecho)
                7 (derecho)
Buscamos 5: ENCONTRADO
Buscamos 1: ENCONTRADO
Buscamos 6: ENCONTRADO
Buscamos 3: ENCONTRADO
Buscamos 7: ENCONTRADO
Buscamos 2: ENCONTRADO
Buscamos 4: ENCONTRADO
Buscamos 4: ENCONTRADO
Buscamos 2: ENCONTRADO
Buscamos 7: ENCONTRADO
Buscamos 3: ENCONTRADO
Buscamos 6: ENCONTRADO
Buscamos 1: ENCONTRADO
Buscamos 5: ENCONTRADO
```