**Proyecto Segundo Parcial - 2011**

**Algoritmos y Estructuras de Datos**

**Árbol de Bayer**



Integrantes:

Murphy, Vegas

Nazzi, Fabio

Carballo, Federico

Consigna:

Árbol de Bayer (B-Tree)

Deberá completar las funciones faltantes y corregir las que están implementadas solo a los fines de prueba.

La inserción en un nodo deberá ser ordenada, preferentemente utilizando ordenación por inserción. La búsqueda en un nodo deberá ser binaria.

Para la correcta división de los nodos deberá auxiliarse con una pila.

Deberá mostrar un listado con los datos del árbol leyéndolos en forma ordenada desde el mismo.

Desarrollo:

La idea tras los árboles-B es que los nodos internos deben tener un número variable de nodos hijo dentro de un rango predefinido. Cuando se inserta o se elimina un dato de la estructura, la cantidad de nodos hijo varía dentro de un nodo. Para que siga manteniéndose el número de nodos dentro del rango predefinido, los nodos internos se juntan o se parten. Dado que se permite un rango variable de nodos hijo, los árboles-B no necesitan rebalancearse tan frecuentemente como los [árboles binarios de búsqueda auto-balanceables](http://es.wikipedia.org/wiki/%C3%81rbol_binario_de_b%C3%BAsqueda_auto-balanceable), pero por otro lado pueden desperdiciar memoria, porque los nodos no permanecen totalmente ocupados. Los límites superior e inferior en el número de nodos hijo son definidos para cada implementación en particular. Un árbol-B se mantiene balanceado porque requiere que todos los nodos hoja se encuentren a la misma altura.

Para implementar el Árbol-B utilizamos 4 clases diferentes: Lista1, Lista, NodoArbolB y ArbolB.

La clase Lista se utilizó para implementar una pila que contiene datos de tipo puntero a NodoArbolB con el objetivo de ir guardando los nodos padres a medida que se recorre el árbol hacia abajo. Es decir, se guarda siempre el Nodo anterior al que se está utilizando.

La clase Lista1 se utilizó para crear una lista de datos tipo entero para pasar los elementos que contiene el ArbolB a una lista.

La clase NodoArbolB es la clase que forma los nodos que luego se entrelazan en el ArbolB. En en ella se almacenan: un arreglo de datos enteros y un arreglo de punteros a otros nodos de la clase NodoArbolB. El tamaño del arreglo de datos es de Orden+1 (para poder insertar provisoriamente el dato que provoca la división del nodo, antes de dividirlo); el tamaño del arreglo de puntero es de Orden+2 por la misma razón, y ya que siempre se tiene un nodo hijo más de la cantidad de datos que hay almacenado. Esta clase también incluye las funciones para Insertar datos, ordenándolos por el método de inserción, y los métodos para obtener la ubicación de un dato y saber si un dato se encuentra o no en el arreglo, realizando una búsqueda binaria.

La clase ArbolB es la encargada de entrelazar a todos los nodos. Contiene un puntero al NodoArbolB raíz, es decir el primer nodo. Además contiene las funciones necesarias para calcular la altura del árbol, insertar un dato, imprimir ordenadamente el Árbol y pasar los datos del árbol a una lista.

Para calcular la altura del árbol se aprovecha del hecho que éste siempre se encuentra balanceado y por lo tanto se recorre siempre por el lado izquierdo, hasta llegar a una hoja contando la cantidad de nodos recorridos.

Para insertar un dato, primero se recorre el árbol hasta llegar a una hoja. Luego se analiza si esa hoja ya está llena o no. Si no lo está, se inserta directamente; caso contrario, se inserta provisoriamente y luego se realiza la división. Luego se realiza otra llamada a la función para insertar en el nodo superior el dato y los punteros a los nuevos hijos que deben “subir”.

Para realizar la impresión y pasar a una lista se utilizó prácticamente el mismo procedimiento. Se recorre el árbol hasta llegar a su hoja izquierda que contiene los menores elementos. Éstos se imprimen o se guardan, dependiendo del caso. Luego, se imprime el dato en la primera posición del nodo padre, se procede con la siguiente hoja y el siguiente dato del nodo padre, en forma sucesiva.