Práctica 15: Árboles Binarios de Búsqueda en C++

Teoría.

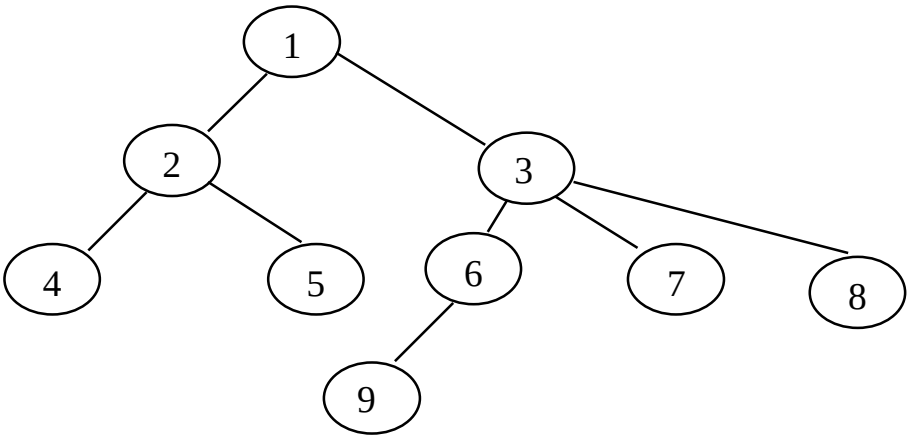
Las estructuras dinámicas vistas hasta ahora son lineales (listas enlazadas, pilas, colas). Estas estructuras tienen grandes ventajas en cuanto a flexibilidad sobre las representaciones contiguas, pero tienen un punto débil: son listas secuenciales, es decir, están dispuestas de forma que es necesario recorrer cada posición al menos una vez (cada elemento tiene un sucesor). Mediante los árboles binarios se introduce el concepto de estructura de bifurcación, donde cada elemento puede tener más de un posible *'siguiente'*, con lo cual es posible resolver algunos problemas de difícil solución cuando se usan estructuras dinámicas lineales.

El árbol es una estructura muy usada en todos los ámbitos de la informática ya que se adapta a la representación natural de informaciones homogéneas organizadas y de una gran comodidad y rapidez de manipulación. Las estructuras tipo árbol se usan para representar datos con una relación jerárquica entre sus elementos, como son árboles genealógicos, tablas, etc.

Un árbol se define como un conjunto finito de uno o más nodos relacionados de la siguiente forma:

- Hay un nodo especial llamado **raíz** del árbol, que proporciona un punto de entrada a la estructura.
- Los nodos restantes se subdividen en m>=0 conjuntos disjuntos, cada uno de los cuales es a su vez un árbol. Estos árboles se llaman **subárboles** del raíz.

Nótese que esta definición es recursiva, se define un árbol en función de otros árboles.
La representación y terminología de los árboles se realiza con las típicas notaciones de las relaciones familiares en los árboles genealógicos: padre, hijo, hermano, ascendiente, descendiente. Junto a estos conceptos se definen otros tales como raíz, nodo, hoja, camino, nivel, profundidad, etc., con los cuales se supone familiarizado al alumno.



Un Árbol Binario se define como un conjunto de 0 ó más nodos tales que:

- Existe un nodo llamado raíz del árbol.
- Cada nodo puede tener 0,1, ó 2 subárboles conocidos como subárbol izquierdo y subárbol derecho.

Un árbol binario puede ser representado fácilmente eligiendo las estructuras de datos adecuadas. Un árbol general puede transformarse en un árbol binario aplicando determinados algoritmos de conversión, el resto del epígrafe se centrará en el estudio de los árboles binarios.
La representación dinámica de un árbol binario utiliza variables punteros y asignación dinámica de espacios de memoria. Cada nodo del árbol contiene al menos los siguientes campos:

- Campo de datos que almacena el tipo de datos.
- Puntero al subárbol izquierdo.
- Puntero al subárbol derecho.

Se llama **recorrido de un árbol** binario al proceso que permite acceder una sola vez a cada uno de los nodos del árbol. Cuando un árbol se recorre, el conjunto completo de nodos se examina. Los algoritmos de recorrido de un árbol realizan las siguientes tareas comunes:

- Procesar el nodo raíz.
- Recorrer el subárbol izquierdo.
- Recorrer el subárbol derecho.

El orden en que se realizan estas acciones da nombre a los tres algoritmos más usuales: pre-orden, in-orden y post-orden:

```
void inOrden(TArbol a)
{ if (!arbolVacio(a))
  { inOrden(Izda(a));
    procesar(a);
    inOrden(Decha(a));
  }
}
```

```
void preOrden(TArbol a)
{ if (!arbolVacio(a))
  { procesar(a);
    preOrden(Izda(a));
    preOrden(Decha(a));
  }
}
```

```
void postOrden(TArbol a)
{ if (!arbolVacio(a))
  { postOrden(Izda(a));
    postOrden(Decha(a));
    procesar(a);
  }
}
```

En cuanto a las operaciones aplicables a un árbol binario se muestran en el módulo de implementación `Marbol` las siguientes:

- `crearVacio`: Devuelve un árbol binario vacío.
- `crearRaiz`: Crea el nodo raíz de un árbol, y almacena en este nodo el valor que se le pasa como parámetro a la función.
- `crearIzda`: Dado un árbol binario que no tiene desarrollada su rama izquierda, esta función crea un nodo con el valor que se recibe como argumento, y se asigna a la rama izquierda del árbol binario original.
- `crearDcha`: Es similar al procedimiento anterior pero aplicable a la rama derecha.
- `sacarRaiz`: Esta función devuelve el valor de la raíz de un árbol binario.
- `arbolVacio`: Determina si un árbol binario está o no vacío.
- `destruirArbol`. Libera la memoria utilizada por el árbol. Hace un recorrido en postorden.

Árboles binarios de búsqueda: es un árbol binario en el que el subárbol izquierdo de cualquier nodo (si no está vacío) contiene valores menores que el que contiene dicho nodo, y el subárbol derecho (si no está vacío) contiene valores mayores.

Práctica

Se desea realizar un programa para llevar la contabilidad de existencias de revistas en un kiosco. De cada revista, se almacenará su nombre y el número de unidades existentes de la misma. Para una mayor eficiencia, se decide usar un árbol binario de búsqueda. El programa deberá presentar el siguiente menú:

```
MENU KIOSCO REVISTAS
--------------------
Autor: <Apellidos> <Nombre>
A. Inserta una revista en el kiosko.
B. Busca una revista en el kiosko.
C. Eliminar una revista del kiosko.
D. Modifica una revista del kiosko.
E. Listar revistas kiosko.
X. Salir del Programa.

Introduzca Opci¾n:
```