Árboles

Introducción

Un **árbol** (*tree*) es una estructura no lineal que se presenta tanto en la vida cotidiana como en la ciencia de la computación: árboles genealógicos, árbol de búsqueda, árbol de juegos, un organigrama. Es una estructura no lineal que parte de un primer elemento, llamado raíz, y luego se "ramifica" en dos o más líneas. Un árbol es un conjunto de nodos y líneas (grafo). Cada nodo contiene información.

Características de los árboles

Los árboles tienen las siguientes propiedades:

- Existe un nodo distinguido llamado **raíz**.
- Todos los nodos, excepto la raíz, tienen una sola línea de entrada.
- Si hay una ruta de **a** a **b**, entonces **b** es hijo de **a**.
- Los nodos que no tienen hijos (nodos terminales) se llaman nodos **hoja**.
- El **grado** de un nodo es la cantidad de hijos que tiene.
- La fila en que reside un nodo es su **nivel**.
- La **altura** de un árbol es la cantidad de líneas en una ruta que empieza en la raíz y termina en el nodo más lejano. Un árbol que solo tiene un nodo (el raíz) tiene *altura* 0.

Tipos de árboles

Hay distintos tipos de árbol:

- **Binario**: todos sus nodos tienen como máximo dos hijos.
  - binario de búsqueda.
  - binario de expresión.
  - binario equilibrado.
  - AVL.
- **Árbol B**: pueden tener tres o más hijos cada nodo.
  - Árbol B*

- Árbol B+

En este módulo en particular, solo estudiaremos los **árboles binarios de búsqueda**.

Los árboles binarios tienen, por cada nodo, como máximo dos hijos a los que llamaremos ***subárbol izquierdo*** y ***subárbol derecho.***

Un **árbol binario de búsqueda** es aquel en el que, para cualquier nodo, su valor es mayor que los valores de los nodos en su subárbol izquierdo y menor que los valores de los nodos en el subárbol derecho.

Implementación mediante listas simplemente enlazadas.

Los árboles binarios se pueden implementar con arreglos o con estructuras enlazadas con punteros. Realizaremos esta última implementación.

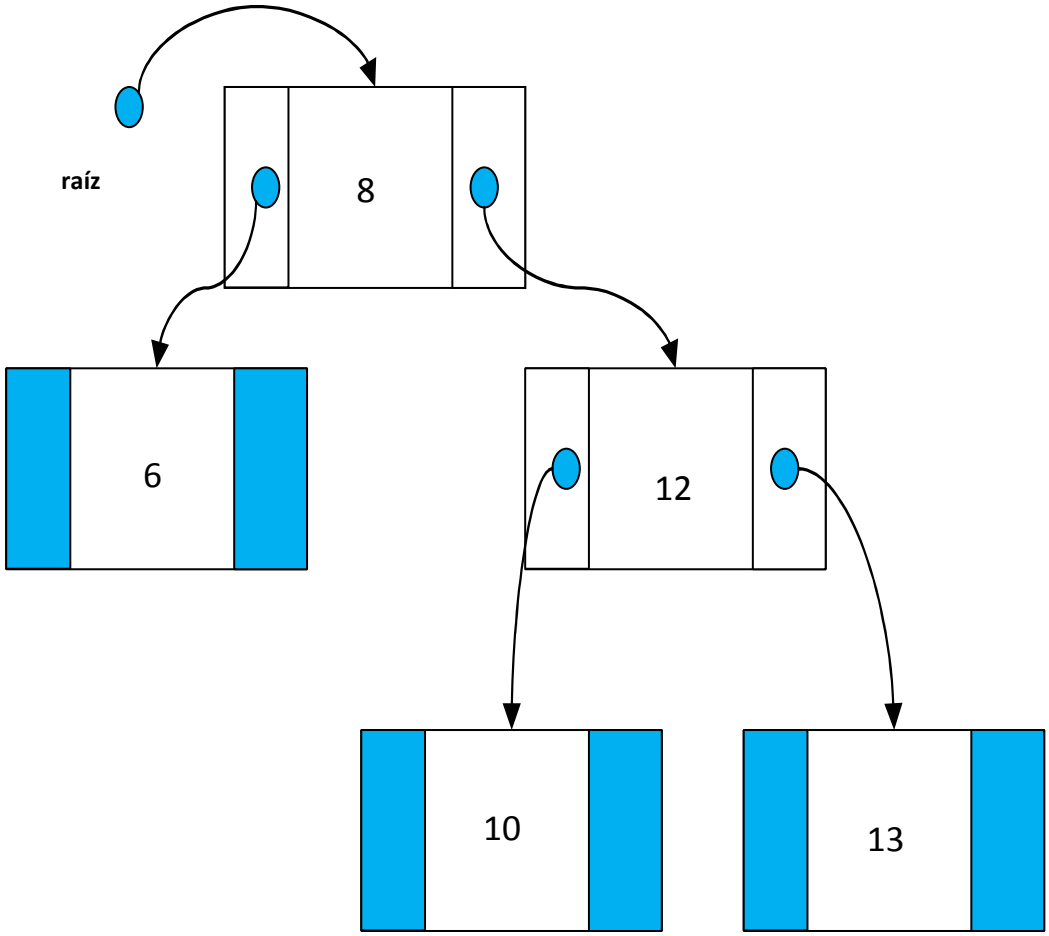
Cada nodo tendrá la siguiente estructura:

```
typedef struct Arbol{
    int clave;
     struct nodoArbol *izq;
     struct nodoArbol *der;
} *nodoarbol;
```

Con respecto al recorrido, como es una estructura no lineal, hay distintas maneras de recorrer el árbol:

- en orden:
  - Se recorre el subárbol izquierdo.
  - Se lee la raíz.
  - Se recorre el subárbol derecho.
- en preorden:
  - Se lee la raíz.
  - Se recorre el subárbol izquierdo.
  - Se recorre el subárbol derecho.

- en postorden:
  - Se recorre el subárbol izquierdo.
  - Se recorre el subárbol derecho.
  - Se lee la raíz.



Ingresó: 8, 6, 12, 10, 13

Pre order: 8, 6, 12, 10, 13

In order: 6, 8, 10, 12, 13

Post order: 6, 10, 13, 12, 8

Muy importante: El orden en que se ingresan los valores determina la forma del árbol.