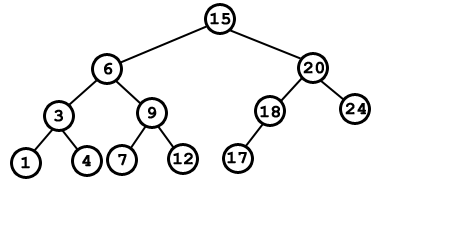
**Concepto de Arboles**

**1. Introducción**

Los árboles en C++ son una estructura de datos fundamental que se utiliza para organizar y almacenar datos de manera jerárquica. Se componen de nodos interconectados, donde cada nodo puede tener cero o más nodos hijos, excepto el nodo superior, conocido como raíz, que no tiene nodos padre. Esta estructura de datos refleja la relación de padres e hijos de una manera natural y se utiliza en una amplia gama de aplicaciones, desde la organización de datos en bases de datos hasta la implementación de algoritmos de búsqueda y ordenación.



*Imagen 1: Árbol básico*

**2. Características:**

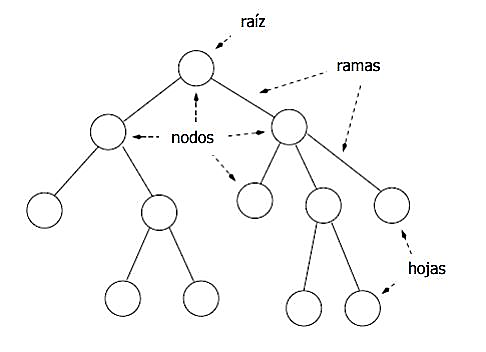
**Jerarquía y estructura recursiva:** Los árboles tienen una estructura jerárquica donde cada nodo puede tener cero o más nodos hijos. Esta estructura se define de forma recursiva, ya que cada nodo puede ser considerado como la raíz de un subárbol que incluye a sus propios hijos y subárboles.

**Nodo raíz:** Es el nodo superior del árbol y actúa como punto de partida para acceder a todos los demás nodos. Este nodo no tiene un nodo padre.

**Nodos internos y hojas:** Los nodos internos son aquellos que tienen al menos un hijo, mientras que las hojas son nodos que no tienen hijos. Los nodos internos se utilizan para estructurar y organizar los datos, mientras que las hojas contienen la información real o los valores almacenados.

**Grado de un nodo:** Se refiere al número de hijos que tiene un nodo. El grado de un nodo puede ser cero (en el caso de las hojas), uno, dos o más.

**Profundidad y altura:** La profundidad de un nodo en un árbol es la longitud del camino desde la raíz hasta ese nodo. La altura de un árbol es la longitud máxima de todos los caminos desde la raíz hasta las hojas. Estas métricas son útiles para medir la eficiencia de ciertas operaciones en un árbol, como la búsqueda.



*Imagen 1: Características del Árbol*

**3. Ejemplo:**

***NodoArbol.h***

#ifndef NODOARBOL\_H

#define NODOARBOL\_H

**class** **NodoArbol** {

**private:**

**int** dato;

NodoArbol\* izquierdo;

NodoArbol\* derecho;

**public:**

NodoArbol(**int** valor);

**int** **obtenerDato**();

**void** **establecerDato**(**int** valor);

NodoArbol\* **obtenerIzquierdo**();

**void** **establecerIzquierdo**(NodoArbol\* nodo);

NodoArbol\* **obtenerDerecho**();

**void** **establecerDerecho**(NodoArbol\* nodo);

};

#endif

***NodoArbol.cpp***

#include "NodoArbol.h"

NodoArbol::NodoArbol(**int** valor) : dato(valor), izquierdo(nullptr), derecho(nullptr) {}

**int** NodoArbol::obtenerDato() {

**return** dato;

}

**void** NodoArbol::establecerDato(**int** valor) {

dato = valor;

}

NodoArbol\* NodoArbol::obtenerIzquierdo() {

**return** izquierdo;

}

**void** NodoArbol::establecerIzquierdo(NodoArbol\* nodo) {

izquierdo = nodo;

}

NodoArbol\* NodoArbol::obtenerDerecho() {

**return** derecho;

}

**void** NodoArbol::establecerDerecho(NodoArbol\* nodo) {

derecho = nodo;

}

***ArbolBinario.h***

#ifndef ARBOLBINARIO\_H

#define ARBOLBINARIO\_H

#include "NodoArbol.h"

**class** **ArbolBinario** {

**private:**

NodoArbol\* raiz;

**void** **destruirArbol**(NodoArbol\* nodo);

**public:**

ArbolBinario();

~ArbolBinario();

**void** **insertar**(**int** valor);

};

#endif

***ArbolBinario.cpp***

#include "ArbolBinario.h"

ArbolBinario::ArbolBinario() : raiz(nullptr) {}

ArbolBinario::~ArbolBinario() {

destruirArbol(raiz);

}

**void** ArbolBinario::destruirArbol(NodoArbol\* nodo) {

**if** (nodo) {

destruirArbol(nodo->obtenerIzquierdo());

destruirArbol(nodo->obtenerDerecho());

**delete** nodo;

}

}

**void** ArbolBinario::insertar(**int** valor) {

}

***main.cpp***

#include <iostream>

#include "ArbolBinario.h"

**int** **main**() {

ArbolBinario arbol;

arbol.insertar(**5**);

arbol.insertar(**3**);

arbol.insertar(**7**);

**return** **0**;

}

**4. Conclusiones**

Los árboles son una estructura de datos fundamental en informática, caracterizada por su organización jerárquica donde cada nodo puede tener cero o más nodos hijos, excepto la raíz, que no tiene nodos padre. Su versatilidad y amplia aplicabilidad se reflejan en su uso en diversos campos, desde la representación de datos en bases de datos y sistemas de archivos hasta la implementación de algoritmos de búsqueda y ordenación.

**5. Referencias**

* Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to Algorithms (4th ed.). MIT Press.

<https://dl.ebooksworld.ir/books/Introduction.to.Algorithms.4th.Leiserson.Stein.Rivest.Cormen.MIT.Press.9780262046305.EBooksWorld.ir.pdf>

* Drozdek, Adam. "Data Structures and Algorithms in C++." 5th Edition, Cengage Learning, 2019.

<https://itlectures.ro/wpcontent/uploads/2016/04/AdamDrozdek__DataStructures_and_Algorithms_in_C_4Ed.pdf>