**Arbol Binario**

se va ordenando según ingresen los números esto dependerá del valor de la raíz se comprobando con la raíz los números que ingresan y se van a la izquierda si es menor a la raíz y a la derecha si es mayor que la raíz.

**Código:**

#include <iostream>

#include <stdlib.h>

#include <windows.h>

using namespace std;

/\*--------- Estructura del arbol -------\*/

typedef class nodo {

public:

int nro;

nodo\* izq, \* der;

}\*ABB;

int numNodos = 0; // nummero de nodos del arbol ABB

int numK = 0, k; // nodos menores que un numero k ingresado

/\* ---------- Estructura de la cola ---------\*/

class nodoCola {

public:

ABB ptr;

nodoCola\* sgte;

};

class cola {

public:

nodoCola\* delante;

nodoCola\* atras;

};

void inicializaCola(cola& q) {

q.delante = NULL;

q.atras = NULL;

}

void encola(cola& q, ABB n) {

nodoCola\* p;

p = new nodoCola();

p->ptr = n;

p->sgte = NULL;

if (q.delante == NULL)

q.delante = p;

else

(q.atras)->sgte = p;

q.atras = p;

}

ABB desencola(struct cola& q) {

nodoCola\* p;

p = q.delante;

ABB n = p->ptr;

q.delante = (q.delante)->sgte;

delete(p);

return n;

}

ABB crearNodo(int x) {

ABB nuevoNodo = new(nodo);

nuevoNodo->nro = x;

nuevoNodo->izq = NULL;

nuevoNodo->der = NULL;

return nuevoNodo;

}

void insertar(ABB& arbol, int x) {

if (arbol == NULL) {

arbol = crearNodo(x);

cout << "\n\t Insertado..!" << endl << endl;

}

else if (x < arbol->nro)

insertar(arbol->izq, x);

else if (x > arbol->nro)

insertar(arbol->der, x);

}

void preOrden(ABB arbol) {

if (arbol != NULL) {

cout << arbol->nro << " ";

preOrden(arbol->izq);

preOrden(arbol->der);

}

}

void enOrden(ABB arbol) {

if (arbol != NULL) {

enOrden(arbol->izq);

cout << arbol->nro << " ";

enOrden(arbol->der);

}

}

void postOrden(ABB arbol) {

if (arbol != NULL) {

enOrden(arbol->izq);

enOrden(arbol->der);

cout << arbol->nro << " ";

}

}

void gotoxy(int x, int y) {

HANDLE hCon;

COORD dwPos;

dwPos.X = x;

dwPos.Y = y;

hCon = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleCursorPosition(hCon, dwPos);

}

void impresionArbol(ABB arbol, int auxY) {

if (arbol == NULL)

return;

auxX += 5;

impresionArbol(arbol->izq, auxY + 2);

gotoxy(10 + auxX - auxY, 15 + auxY);

cout << arbol->nro << endl << endl;

impresionArbol(arbol->der, auxY + 2);

}

bool busquedaRec(ABB arbol, int dato) {

int r = 0; // 0 indica que no lo encontre

if (arbol == NULL)

return r;

if (dato < arbol->nro)

r = busquedaRec(arbol->izq, dato);

else if (dato > arbol->nro)

r = busquedaRec(arbol->der, dato);

else

r = 1; // son iguales, lo encontre

return r;

}

ABB unirABB(ABB izq, ABB der) {

if (izq == NULL) return der;

if (der == NULL) return izq;

ABB centro = unirABB(izq->der, der->izq);

izq->der = centro;

der->izq = izq;

return der;

}

void elimina(ABB& arbol, int x) {

if (arbol == NULL) return;

if (x < arbol->nro)

elimina(arbol->izq, x);

else if (x > arbol->nro)

elimina(arbol->der, x);

else {

ABB p = arbol;

arbol = unirABB(arbol->izq, arbol->der);

delete p;

}

}

void menu() {

//system("cls");

cout << "\n\t\t ..[ ARBOL BINARIO DE BUSQUEDA ].. \n\n";

cout << "\t [1] Insertar elemento \n";

cout << "\t [2] Mostrar arbol \n";

cout << "\t [3] Recorridos de profundiad \n";

cout << "\t [4] Buscar elemento \n";

cout << "\t [5] Eliminar elemento \n";

cout << "\t [6] SALIR \n";

cout << "\n\t Ingrese opcion : ";

}

void menu2() {

//system("cls"); // para limpiar pantalla

cout << endl;

cout << "\t [1] En Orden \n";

cout << "\t [2] Pre Orden \n";

cout << "\t [3] Post Orden \n";

cout << "\n\t Opcion : ";

}

int main() {

ABB arbol = NULL;

int x;

int op, op2;

system("color f9"); // poner color a la consola

do

{

system("cls");

menu(); cin >> op;

cout << endl;

switch (op)

{

case 1:

cout << " Ingrese valor : "; cin >> x;

insertar(arbol, x);

system("pause");

break;

case 2:

impresionArbol(arbol, 0);

system("pause");

break;

case 3:

menu2(); cin >> op2;

if (arbol != NULL)

{

switch (op2)

{

case 1:

enOrden(arbol); break;

case 2:

preOrden(arbol); break;

case 3:

postOrden(arbol); break;

}

}

else

cout << "\n\t Arbol vacio..!";

system("pause");

break;

case 4:

bool band;

cout << " Valor a buscar: "; cin >> x;

band = busquedaRec(arbol, x);

if (band == 1)

cout << "\n\tEncontrado...";

else

cout << "\n\tNo encontrado...";

system("pause");

break;

case 5:

cout << " Valor a eliminar: "; cin >> x;

elimina(arbol, x);

cout << "\n\tEliminado..!";

system("pause");

break;

case 6:

break;

default:

cout << "Ingrese una opcion correcta" << endl;

system("pause");

break;

}

cout << "\n\n\n";

//system("pause"); // hacer pausa y presionar una tecla para continuar

} while (op != 6);

cout << "Gracias por utilizar :)" << endl;

exit(1);

return (0);

}