29-11-2021

Examen árbol binario Búsqueda

Materia: Estructura de datos

Sección: D01.

Código: 216584703

Carrera: Ingeniería en computación.

Nombre alumno: Padilla Pérez Jorge Daray

Nombre profesor: Julio Esteban Valdes Lopez





Introducción

Mi práctica consiste en la implementación de un árbol binario de busqueda en la cual se implementan lo típico de un TDA arbol lo que viene siendo insertar los nodos del arbol En esta práctica no se logró hacer todo lo pedido para esta, además de no poder realizar la función de localizar posición por elemento la cual me costó más, y no se logró.

También aunque no se implementaron las funciones eliminar y anular creo que es una función difícil de comprender y de hacer espero que próximamente con mas calma pueda realizar de mejor forma los trabajos que se piden.

Pantallazos

Texto

Descripción generada automáticamente

Aquí se muestra la inserción de 3 numeros que se muestran pre orden se inserto el 5 luego el 2 y por ultimo el 7.

Texto

Descripción generada automáticamente

Aquí se muestra el recorrido en orden y el recorrido en posorden.

Conclusión

Respecto a la realización del código concluyo que no se logró el código de la forma esperada por motivos propios y más que nada por el tiempo ya que no me ajusto el tiempo para poder desarrollarlo en buenas condiciones además de no poder terminarlo se pudo aprender mucho de este ultimo que espero poder estudiar en las vacaciones con mas calma ya que se tranquilice mi vida.

También espero poder dearrollar mejores programas con los conocimientos que pude adquirir en esta materia me siento mejor preparado y comprendo mejor las técnicas con las que poder realizar programas mas difíciles es simplemente cuestión de pensar y como se enseño en la clase poder realizar dibujos ayuda mucho para comprender el tema a resolver.

**Código fuente**

#include <iostream>

#include<stdlib.h>

#include <string.h>

#define TIPO\_DATO int

using namespace std;

class Nodo{

protected:

Nodo \* raiz; //ancla

TIPO\_DATO elem;

Nodo \* hijo\_izq;

Nodo \* hijo\_der;

public:

Nodo(){

raiz = nullptr;

};

Nodo(TIPO\_DATO elem);

void insertar(TIPO\_DATO dato);

void rec\_preorden(int nivel);

void rec\_inorden(int nivel);

void rec\_posorden(int nivel);

void rec\_izq(TIPO\_DATO dato, int nivel);

void rec\_der(TIPO\_DATO dato, int nivel);

};

Nodo::Nodo(TIPO\_DATO elem)

{

this->elem = elem;

hijo\_der = nullptr;

hijo\_izq = nullptr;

}

void Nodo::insertar(TIPO\_DATO dato)

{

if( dato < elem){

if(hijo\_izq == nullptr){

hijo\_izq = new Nodo(dato);

} else {

hijo\_izq->insertar(dato);

}

} else {

if(hijo\_der == nullptr){

hijo\_der = new Nodo(dato);

} else {

hijo\_der->insertar(dato);

}

}

}

void Nodo::rec\_preorden(int nivel)

{

for (int i = 0 ; i < nivel ; i++) {

std::cout << " ";

}

std::cout << elem << std::endl;

if(hijo\_izq !=nullptr){

hijo\_izq->rec\_preorden(nivel+1);

}

if(hijo\_der !=nullptr){

hijo\_der->rec\_preorden(nivel+1);

}

}

void Nodo::rec\_inorden(int nivel)

{

for (int i = 0 ; i < nivel ; i++) {

std::cout << " ";

}

if(hijo\_izq !=nullptr){

hijo\_izq->rec\_inorden(nivel+1);

}

std::cout << elem << std::endl;

if(hijo\_der !=nullptr){

hijo\_der->rec\_inorden(nivel+1);

}

}

void Nodo::rec\_posorden(int nivel)

{

for (int i = 0 ; i < nivel ; i++) {

std::cout << " ";

}

if(hijo\_izq !=nullptr){

hijo\_izq->rec\_posorden(nivel+1);

}

if(hijo\_der !=nullptr){

hijo\_der->rec\_posorden(nivel+1);

}

std::cout << elem << std::endl;

}

void Nodo::rec\_izq(TIPO\_DATO dato, int nivel)

{

if(hijo\_izq !=nullptr){

hijo\_izq->rec\_izq(dato, nivel+1);

for (int i = 0 ; i < nivel ; i++) {

std::cout << " ";

}

}

}

void Nodo::rec\_der(TIPO\_DATO dato, int nivel)

{

if(hijo\_der !=nullptr){

hijo\_der->rec\_der(dato, nivel+1);

for (int i = 0 ; i < nivel ; i++) {

std::cout << " ";

}

}

}

class Arbol: Nodo{

public:

bool vacia();

void insertar(TIPO\_DATO dato);

void rec\_preorden();

void rec\_inorden();

void rec\_posorden();

void Localiza\_elemento(TIPO\_DATO dato);

void rec\_izq();

void rec\_der();

};

bool Arbol::vacia()

{

return raiz == nullptr;

}

void Arbol::insertar(TIPO\_DATO dato)

{

if(this->vacia()){

raiz = new Nodo(dato);

} else {

raiz->insertar(dato);

}

}

void Arbol::rec\_preorden()

{

if(not this->vacia()){

raiz->rec\_preorden(0);

}

}

void Arbol::rec\_inorden()

{

if(not this->vacia()){

raiz->rec\_inorden(0);

}

}

void Arbol::rec\_posorden()

{

if(not this->vacia()){

raiz->rec\_posorden(0);

}

}

void Arbol::Localiza\_elemento(TIPO\_DATO dato)

{

if(not this->vacia())

{

if(dato == this->elem)

{

return;

}

else if(dato > this->elem)

{

raiz->rec\_izq(dato, 0);

}

else if (dato < this->elem)

{

raiz->rec\_der(dato, 0);

}

}

else

{

printf("la lista esta vacia");

}

}

void menu()

{

int opc;

Arbol mi\_arbolito;

int dato;

do{

printf("\tINTRODUCE AN OPCION:\n");

printf("0.-Exit\n1.-Arbol binario\n");

scanf("%d", &opc);

switch(opc){

case 0: printf("\tGoodbye");

system("pause");

break;

case 1: printf("\tArbol binario\n\n");

int opcion;

do{

printf("\tINTRODUCE AN OPCION:\n");

printf("0.-Exit\n1.-Insert\n2.-Rec\_preorden\n3.-Rec\_inorden\n4.-Rec\_posorden\n5.-Localiza\n");

scanf("%d", &opcion);

switch(opcion){

case 0: printf("\t Adios\n");

system("pause");

break;

case 1: printf("\tInsertar dato\n\n");

printf("Dato a insertar:\n");

scanf("%d", &dato);

mi\_arbolito.insertar(dato);

break;

case 2: printf("\tRecorrer en preorden\n\n");

mi\_arbolito.rec\_preorden();

break;

case 3: printf("\tRecorrer en orden\n\n");

mi\_arbolito.rec\_inorden();

break;

case 4: printf("\tRecorrer en posorden\n\n");

mi\_arbolito.rec\_posorden();

break;

case 5: printf("\tLocalizar posicion\n\n");

printf("Dato a buscar:\n");

scanf("%d", &dato);

mi\_arbolito.Localiza\_elemento(dato);

break;

/\*case 6: printf("\tInsert first\n\n");

mi\_lista.getinsertlast();

break;

case 7: printf("\tInsert first\n\n");

mi\_lista.getLocaliza();

break;

case 8: printf("\tInsert first\n\n");

mi\_lista.getAnula();

break;\*/

}

}while(opcion!=0);

break;

default: printf("\tChoose a correct value");

break;

}

system("cls");

}while(opc!=0);

}

int main()

{

menu();

return 0;

}