19-5-2022

Practica 6 árbol

Materia: Seminario de estructura de datos 1

Sección: D13.

Código: 216584703

Carrera: Ingeniería en computación.

Nombre alumno: Padilla Pérez Jorge Daray

Nombre profesor: Julio Esteban Valdes Lopez



**Introducción**

En esta practica se realizó la implementación de un árbol binario, en el cual se implementa lo típico del TDA árbol el cual incluye el mostrar de las 3 formas el árbol, ya sea en orden, posorden y preorden se trato de imitar la impresión de los arboles del cmd y mas o menos se consiguió el parecido.

También se añadió la función de buscar el cual encuentra el nodo en el árbol y lo muestra en la pantalla lastimosamente no se pudo completar el árbol ya que falto la función de eliminar de las 3 formas que hay además de la función de anular el árbol.

Texto

Descripción generada automáticamente

Aquí se aprecia el menú del árbol.

Texto

Descripción generada automáticamente

Aquí insertamos varios nodos y se muestra de manera horizontal el arbol.

Texto

Descripción generada automáticamente

Aquí lo imprimimos en preorden.

Texto

Descripción generada automáticamente

Inorden

Texto

Descripción generada automáticamente

Y la impresión en pos orden

**Conclusión**

Se pudo completar de manera correcta el programa, utilizando una clase nodo y otra llamda árbol, en la que se almacenan los nodos, en la cual pues se tienen los métodos principales de un arbol menos los de elimina que falla y el de anular el arbol.

También se desarrolló la impresión del árbol como lo hace el panel de comandos de Windows que se ve bien, no es idéntico pero le hace el parecido ya que esta basado en su impresión.

**Codigo fuente**

#include <iostream>

#define TIPO\_DATO int

using namespace std;

class Nodo{

private:

TIPO\_DATO elem;

Nodo \* hijo\_izq;

Nodo \* hijo\_der;

Nodo \* padre;

public:

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 mostrar\_arbol(int nivel);

int localizar\_arbol(int nivel);

void eliminar\_nodo(int nivel);

void eliminar();

int minimo();

};

Nodo::Nodo(TIPO\_DATO elem)

{

this->elem = elem;

hijo\_der = nullptr;

hijo\_izq = nullptr;

this->padre = padre;

}

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++) {

if (i == 0){

std::cout << " |";

}

else {

std::cout << " ";

}

}

if (nivel == 0){

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

}

else if (nivel == 1){

std::cout << "\_";

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

}

else{

std::cout << "|";

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)

{

if(hijo\_izq !=nullptr){

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

}

std::cout << " ";

std::cout << elem ;

if(hijo\_der !=nullptr){

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

}

}

void Nodo::rec\_posorden(int nivel)

{

if(hijo\_izq !=nullptr){

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

}

if(hijo\_der !=nullptr){

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

}

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

std::cout << " |";

}

if (nivel == 0){

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

}

else if (nivel == 1){

std::cout << "\_";

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

}

else{

std::cout << "\_";

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

}

}

void Nodo::mostrar\_arbol(int nivel)

{

if(hijo\_der !=nullptr){

hijo\_der->mostrar\_arbol(nivel+1);

}

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

std::cout << " ";

}

std::cout << "\_";

std::cout << elem <<endl;

if(hijo\_izq !=nullptr){

hijo\_izq->mostrar\_arbol(nivel+1);

}

}

int Nodo::localizar\_arbol(int nivel)

{

if (nivel == elem){

std::cout <<elem<<endl;

return elem;

}

else if(nivel < elem){

return hijo\_izq->localizar\_arbol(nivel);

}

else{

return hijo\_der->localizar\_arbol(nivel);

}

}

void Nodo::eliminar\_nodo(int nivel)

{

if (nivel == elem){

eliminar();

}

else if (nivel < elem){

hijo\_izq->eliminar\_nodo(nivel);

}

else{

hijo\_der->localizar\_arbol(nivel);

}

}

int Nodo::minimo()

{

if (hijo\_izq !=nullptr)

{

return hijo\_izq->minimo();

}

else{

return elem;

}

}

void Nodo::eliminar()

{

if (hijo\_izq && hijo\_der)

{

int menor = hijo\_der->minimo();

int aux = menor;

eliminar\_nodo(menor);

elem = aux;

}

}

class Arbol{

private:

Nodo \* raiz; //ancla

public:

Arbol();

bool vacia();

void insertar(TIPO\_DATO dato);

void rec\_preorden();

void rec\_inorden();

void rec\_posorden();

void mostrar\_arbol();

void localizar\_arbol(TIPO\_DATO dato);

void eliminar\_nodo(TIPO\_DATO dato);

};

Arbol::Arbol()

{

raiz = nullptr;

}

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::mostrar\_arbol()

{

if(not this->vacia()){

raiz->mostrar\_arbol(0);

}

}

void Arbol::localizar\_arbol(TIPO\_DATO dato)

{

if(not this->vacia()){

raiz->localizar\_arbol(dato);

}

}

void Arbol::eliminar\_nodo(TIPO\_DATO dato)

{

if(not this->vacia()){

raiz->eliminar\_nodo(dato);

}

}

void menu()

{

int opc;

Arbol miArbolito;

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.-imprimir\_arbol\n6.-Localizar\n7.-eliminar\n");

scanf("%d", &opcion);

miArbolito.mostrar\_arbol();

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);

miArbolito.insertar(dato);

break;

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

miArbolito.rec\_preorden();

std::cout << " |";

std::cout << "\_";

std::cout << "null"<<endl;

break;

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

miArbolito.rec\_inorden();

break;

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

miArbolito.rec\_posorden();

break;

case 5: printf("\tmostrar\_arbol\n\n");

miArbolito.mostrar\_arbol();

break;

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

std::cout << "Ingrese el numero a buscar: "<<endl;

std::cin >> dato;

miArbolito.localizar\_arbol(dato);

break;

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

std::cout << "Ingrese el numero a eliminar: "<<endl;

std::cin >> dato;

miArbolito.eliminar\_nodo(dato);

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;

}