//node.h

#include<iostream>

using namespace std;

class Node

{

Node \*left;

Node \*right;

int data;

public:

Node();

Node(int);

void setData(int);

int getData()const;

void setleft(Node \*);

void setright(Node \*);

Node \*getleft()const;

Node \*getright()const;

};

//node.cpp

#include "pch.h"

#include "node.h"

Node::Node()

{

this->data = 0;

this->left = NULL;

this->right = NULL;

}

Node::Node(int data)

{

this->data = data;

this->left = NULL;

this->right = NULL;

}

void Node::setData(int data)

{

this->data = data;

}

int Node::getData() const

{

return this->data;

}

void Node::setleft(Node \*left)

{

this->left = left;

}

void Node::setright(Node \*right)

{

this->right = right;

}

Node\* Node::getleft() const

{

return this->left;

}

Node\* Node::getright() const

{

return this->right;

}

//tree.h

#pragma once

#include "node.h"

class CTree

{

Node \*root;

public:

CTree();

//CTree(int);

bool insert(int);

void Inorder(Node \*);

void setRoot(Node \*);

void nonrec\_inorder();

void nonrec\_Preorder();

void nonrec\_postorder();

Node \*getRoot();

void delet(int);

};

//tree.cpp

#include "pch.h"

#include "tree.h"

CTree::CTree()

{

Node\* root = NULL;

}

bool CTree::insert(int data)

{

Node \*newNode = new Node(data);

if (newNode == NULL)

return false;

if (root == NULL) {

root = newNode;

return true;

}

Node \*trav = root;

Node \* prev = trav;

while (trav != NULL)

{

prev = trav;

if (data > trav->getData())

trav = trav->getright();

else

trav = trav->getleft();

}

if (data > prev->getData())

prev->setright(newNode);

else

prev->setleft(newNode);

return true;

}

void CTree::Inorder(Node \*root)

{

if (root == NULL)

return;

else

{

Inorder(root->getleft());

cout <<" " <<root->getData();

Inorder(root->getright());

}

}

void CTree::setRoot(Node \*root)

{

this->root = root;

}

Node\* CTree::getRoot()

{

return this->root;

}

void CTree::nonrec\_inorder()

{

Node \*stack[100];

int top = -1;

Node \*temp = root;

while (top != -1 || temp != NULL)

{

while (temp)

{

stack[++top]=temp;

temp = temp->getleft();

}

temp = stack[top--];

cout << temp->getData() << " ";

temp = temp->getright();

}

cout << endl;

}

void CTree::nonrec\_Preorder()

{

Node \*stack[100];

Node \*temp = root;

int top = -1;

while (top != -1 || temp != NULL)

{

while (temp)

{

cout << temp->getData() << " ";

stack[++top]=temp;

temp = temp->getleft();

}

temp = stack[top--];

temp = temp->getright();

}

}

void CTree::nonrec\_postorder()

{

typedef struct

{

Node \* addr;

char flag;

}pair;

pair stk[100];

int top = -1;

Node \*temp = root;

while(temp!=NULL || top!=-1)

{

while (temp)

{

pair p;

p.addr = temp;

p.flag = 'L';

stk[++top] = p;

temp = temp->getleft();

}

pair p = stk[top--];

if (p.flag == 'L')

{

temp = p.addr;

temp = temp->getright();

p.flag = 'R';

stk[++top] = p;

}

else

{

temp = p.addr;

cout << temp->getData() << " ";

temp = NULL;

}

}

cout << endl;

}

void CTree::delet(int data)

{

if ((root->getleft() == NULL) && (root->getright() == NULL))

{

delete root;

root = NULL;

cout << "Deleted...";

}

Node \*trav = root;

Node \*prev = trav;

while (trav ->getleft()!= NULL && trav->getright()!=NULL)

{

prev = trav;

if (data > trav->getData())

trav = trav->getright();

else

trav = trav->getleft();

}

if (data > prev->getData())

{

prev->setright(NULL);

delete trav;

}

else

{

prev->setleft(NULL);

delete trav;

}

}

//binarysrch.cpp

#include "pch.h"

#include "tree.h"

#include <iostream>

using namespace std;

#include<conio.h>

int main()

{

CTree bst;

int choice, data;

//Node \*root;

do

{

cout << "\n1.Insert \n2.Display \n3.nonrec\_Inorder \n4.nonrec\_preorder \n5.Non Rec Postorder \n6.Delete 7.Exit" << endl;

cout << "Enter the choice :" << endl;

cin >> choice;

switch (choice)

{

case 1:

cout << "Enter the data :";

cin >> data;

bst.insert(data);

break;

case 2:

bst.Inorder(bst.getRoot());

break;

case 3:

bst.nonrec\_inorder();

break;

case 4:

bst.nonrec\_Preorder();

break;

case 5:bst.nonrec\_postorder(); break;

case 6:

cout << "Enter data to delete:[leaf]: ";

cin >> data;

bst.delet(data); break;

case 7:exit(0);

default:cout << "Wrong choice .....\n";

}

} while (choice != 7);

\_getch();

return 0;

}