A picture containing text

Description automatically generated



Lab Journal: 9

Date: 23 / 11 / 24

Student : Abdul Rafay

Enrollment : 01-131232-004

Department of Software Engineering

Bahria University, Islamabad

Data Structures & Algorithms Lab (Spring-2024)

Teacher: RAHEELA AMBRIN

**Comments:**

**Signature**

### Code:

All the code files are uploaded on GitHub: <https://github.com/CharlieFour/DSA_Lab>

You can check out the code on GitHub in Lab\_09 folder.

#### Binary Tree ADT:

#include <iostream>  
  
template <class T>  
struct node  
{  
    T info;  
    node \*left, \*right, \*parent;    
};  
  
template <class T>  
using nodeptr = node<T>\*;  
  
template <class T>  
class BinaryTree  
{  
    private:  
        int numNodes;  
        int count;  
    public:  
        nodeptr<T> tree;  
        BinaryTree();  
        BinaryTree(int n);  
        void clear(nodeptr<T> tree);  
        ~BinaryTree();  
        nodeptr<T> makeTree(T info);  
        void insert(T info);  
        void iL(nodeptr<T> node, T info);  
        void iR(nodeptr<T> node, T info);  
        void preTrav(nodeptr<T> tree);  
        void inTrav(nodeptr<T> tree);  
        void postTrav(nodeptr<T> tree);  
};  
  
template <class T>  
BinaryTree<T>::BinaryTree()  
{  
    tree = NULL;  
    numNodes = 500;  
    count = 0;  
}  
  
template <class T>  
void BinaryTree<T>::clear(nodeptr<T> tree)  
{  
    if (tree != NULL)  
    {  
        clear(tree->left);  
        clear(tree->right);  
        delete tree;  
    }  
}  
  
template <class T>  
BinaryTree<T>::~BinaryTree()  
{  
    clear(tree);  
}  
  
template <class T>  
BinaryTree<T>::BinaryTree(int n)  
{  
    tree = NULL;  
    numNodes = n;  
    count = 0;  
}  
  
template <class T>  
nodeptr<T> BinaryTree<T>::makeTree(T info)  
{  
    nodeptr<T> p = new node<T>;  
    p->info = info;  
    p->left = NULL;  
    p->right = NULL;  
    return p;  
}  
  
template <class T>  
void BinaryTree<T>::iL(nodeptr<T> node, T info)  
{  
    if(node->left != NULL)  
    {  
        std::cerr << "Invalid Insertion" << std::endl;  
    }  
    else  
    {  
        node->left = makeTree(info);  
    }          
}  
  
template <class T>  
void BinaryTree<T>::iR(nodeptr<T> node, T info)  
{  
    if(node->right != NULL)  
    {  
        std::cerr << "Invalid Insertion" << std::endl;  
    }  
    else  
    {  
        node->right = makeTree(info);  
    }  
}  
  
template <class T>  
void BinaryTree<T>::insert(T info)  
{  
    if (tree == NULL)  
    {  
        tree = makeTree(info);  
    }  
    else  
    {  
        nodeptr<T> p = tree;  
        while (p != NULL)  
        {  
            if (info < p->info)  
            {  
                if (p->left == NULL)  
                {  
                    iL(p, info);  
                    return;  
                }  
                else  
                {  
                    p = p->left;  
                }  
            }  
            else  
            {  
                if (p->right == NULL)  
                {  
                    iR(p, info);  
                    return;  
                }  
                else  
                {  
                    p = p->right;  
                }  
            }  
        }  
    }  
}  
  
template <class T>  
void BinaryTree<T>::preTrav(nodeptr<T> tree)  
{  
    if (tree != NULL)  
    {  
        std::cout << tree->info << " ";  
        preTrav(tree->left);  
        preTrav(tree->right);  
    }  
}  
  
template <class T>  
void BinaryTree<T>::inTrav(nodeptr<T> tree)  
{  
    if (tree != NULL)  
    {  
        inTrav(tree->left);  
        std::cout << tree->info << " ";  
        inTrav(tree->right);  
    }  
}  
  
template <class T>  
void BinaryTree<T>::postTrav(nodeptr<T> tree)  
{  
    if (tree != NULL)  
    {  
        postTrav(tree->left);  
        postTrav(tree->right);  
        std::cout << tree->info << " ";  
    }  
}

#### Main

#include <iostream>  
#include "../lib/binarytree.cpp"  
  
using namespace std;  
  
int main()  
{  
    BinaryTree<int> tree;  
    cout << "Simple binary insertion" << endl;  
    for(int i = 0; i < 10; i++)  
    {  
        int a = rand() % 10;  
        tree.insert(a);  
        cout << a << " ";  
    }  
    cout << "\n\nPre Order" << endl;  
    tree.preTrav(tree.tree);  
    cout << "\n\nIn Order" << endl;  
    tree.inTrav(tree.tree);  
    cout << "\n\nPost Order" << endl;  
    tree.postTrav(tree.tree);  
    cout << endl;  
    system("pause");  
    return 0;  
}

### Screen Shots:

