Bahria University, Islamabad

Department of Software Engineering

Data Structures & Algorithms Lab

(Spring-2024)

Teacher: RAHEELA AMBRIN

Student : Abdul Rafay

Enrollment : 01-131232-004

Lab Journal: 10

Date: 01 / 12 / 24

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_10 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);
        T find(T &info);
};

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>
T BinaryTree<T>::find(T &info)
{
    nodeptr<T> p = tree;
    while (p != NULL)
    {
        if (info == p->info)
        {
            return p->info;
        }
        else if (info < p->info)
        {
            p = p->left;
        }
        else
        {
            p = p->right;
        }
    }
    return -1;
}
```

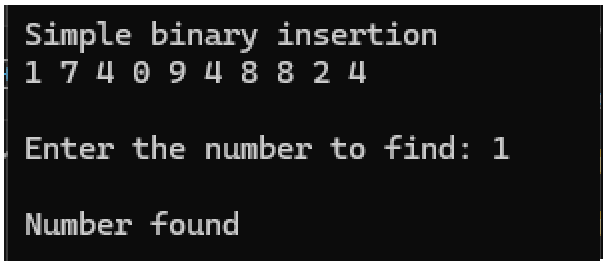
Main

```
#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\nEnter the number to find: ";
    int b;
    cin >> b;
    if (tree.find(b) == b)
    {
        cout << "\nNumber found" << endl;
    }
    else if (tree.find(b) == -1)
    {
        cout << "\nNumber not found" << endl;
    }
    cout << endl;
    system("pause");
    return 0;
}
```

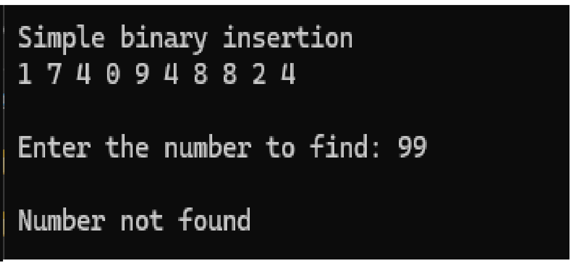
Screen Shots:



```
Simple binary insertion
1 7 4 0 9 4 8 8 2 4

Enter the number to find: 1

Number found
```



```
Simple binary insertion
1 7 4 0 9 4 8 8 2 4

Enter the number to find: 99

Number not found
```