**二叉树实验报告**

13软卓 甘芷翘 201335620066

**1、实验内容**

实验目的：

a. 掌握二叉树的定义和存储表示，学会建立一颗特定的二叉树的方法。

b.掌握二叉树的遍历算法（先序、中序、后序遍历算法）的思想，并学会遍历算法的递归实现

实验题目：

1. 对于给定的一二叉树，实现各种约定的遍历。

构造二叉树，再实现二叉树的先序、中序、后序遍历，最后统计二叉树的深度

1. Page 197-5.6、5.8、5.9

**2、代码及结果展示**

1.

***BSTNode.h***

#ifndef BSTNODE\_H

#define BSTNODE\_H

#include<iostream>

using namespace std;

template <typename E>

class BSTNode

{

private:

E it;

BSTNode\* lc;

BSTNode\* rc;

int count;

public:

BSTNode()

{

lc=rc=NULL;

count=0;

}

BSTNode(E e,BSTNode\* l=NULL,BSTNode\*r=NULL)

{

it=e;

lc=l;

rc=r;

count=0;

}

~BSTNode()

{}

E& element()//get value

{

return it;

}

void setElement(const E& e)//set value

{

it=e;

}

inline BSTNode\* left() const//return left

{

return lc;

}

void setLeft(BSTNode<E>\*b)//set left node

{

lc=(BSTNode\*)b;

}

inline BSTNode\* right() const//return right

{

return rc;

}

void setRight(BSTNode<E>\*b)//set right node

{

rc=(BSTNode\*)b;

}

bool isLeaf()//determine leaf node

{

return (lc==NULL)&&(rc==NULL);

}

void visit(BSTNode<E>\* node)//output node value

{

cout<<node->element();

}

};

#endif

***Tree.h***

#ifndef TREE\_H

#define TREE\_H

#include"BSTNode.h"

template<typename E>

class Tree:public BSTNode<E>

{

public:

Tree()

{

this->root = Creat( );

}

~Tree(void)

{

Release(root);

}

BSTNode<E>\* Getroot()//get root

{

return root;

}

void preorder(BSTNode<E>\* rootS)//visit preorder

{

if(rootS==NULL)

return;

visit(rootS);

preorder(rootS->left());

preorder(rootS->right());

}

void inorder(BSTNode<E>\* rootS)//visit inorder

{

if(rootS==NULL)

return;

inorder(rootS->left());

visit(rootS);

inorder(rootS->right());

}

void postorder(BSTNode<E>\* rootS)//visit postorder

{

if(rootS==NULL)

return;

postorder(rootS->left());

postorder(rootS->right());

visit(rootS);

}

int TreeDepth(BSTNode<E>\* T)//height of tree

{

int leftH,rightH,hl;

if(T==0)

hl=0;

else

{

leftH=TreeDepth(T->left());

rightH=TreeDepth(T->right());

hl=(leftH>rightH?leftH:rightH)+1;

}

return hl;

}

private:

BSTNode<E> \*root;

BSTNode<E> \*Creat()//create new BST

{

BSTNode<E>\* root;

E ch;

cout<<"please enter the node data of BST by preorder(empty: enter ?)"<<endl;

cin>>ch;

if (ch=='?')

root = NULL;

else

{

root = new BSTNode<E>;

root->setElement(ch);

root->setLeft(Creat( ));

root->setRight(Creat( ));

}

return root;

}

void Release(BSTNode<E> \*root)//delete BST

{

if (root != NULL)

{

Release(root->left());

Release(root->right());

delete root;

}

}

};

#endif

***BinaryTree.cpp***

#include"Tree.h"

#include<iostream>

using namespace std;

int main()

{

Tree<char> bt; //create a tree

BSTNode<char>\* root = bt.Getroot( );

cout<<"------preorder------ "<<endl;

bt.preorder(root);

cout<<endl;

cout<<"------inorder------ "<<endl;

bt.inorder(root);

cout<<endl;

cout<<"------postorder------ "<<endl;

bt.postorder(root);

cout<<endl;

cout<<"------depth of tree------ "<<endl;

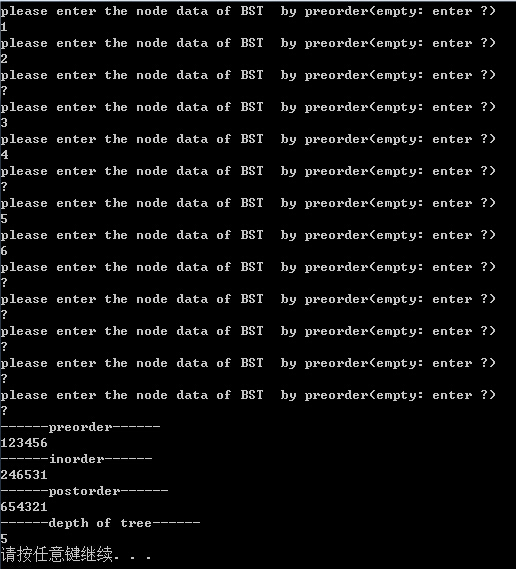
cout<<bt.TreeDepth(root)-1;

cout<<endl;

system("pause");

return 0;

}



2.

(5.6)

***SEARCH Function:***

bool search(BSTNode<E>\*root,E K)

{

if(root->element()==K)

return true;

else

{

if(root->left()!=NULL&&(search(root->left(),K)))

return true;

if(root->right()!=NULL&&(search(root->right(),K)))

return true;

}

return false;

}

***BinaryTree.cpp***

#include"Tree.h"

#include<iostream>

using namespace std;

int main()

{

int x=1;

while(x==1)

{

Tree<char> bt; //create a tree

BSTNode<char>\* root = bt.Getroot( );

cout<<"------preorder------ "<<endl;

bt.preorder(root);

cout<<endl;

cout<<"------inorder------ "<<endl;

bt.inorder(root);

cout<<endl;

cout<<"------postorder------ "<<endl;

bt.postorder(root);

cout<<endl;

cout<<"------depth of tree------ "<<endl;

cout<<bt.TreeDepth(root)-1;

cout<<endl;

int y=1;

while(y==1)//find the value of k

{

cout<<"please input the value K you want to find :";

char K;

cin>>K;

if(bt.search(root,K))

cout<<K<<" is in your binary tree!";

else

cout<<K<<" isn't in your binary tree!";

cout<<endl<<endl;

cout<<"CHOOSE:"<<endl

<<"1.Build new binary tree"<<endl

<<"2.Search new value"<<endl

<<"3.Quit"<<endl

<<"Answer:";

int choose;

cin>>choose;

if(choose==1)//choose keep looking or build new tree or quit

{

x=1;

y=2;

system("cls");

}

if(choose==2)

{

y=1;

}

if(choose==3)

{

x=2;

y=2;

}

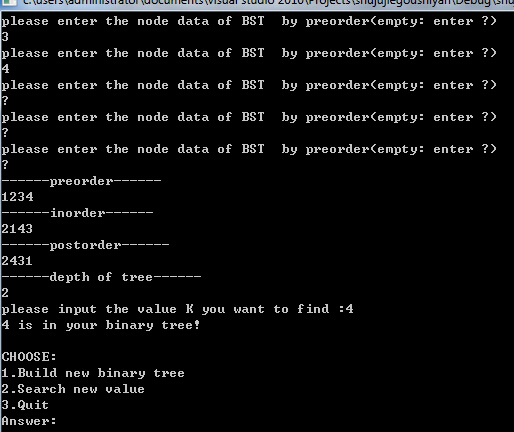
}

}

system("pause");

return 0;

}



(5.8)

int TreeDepth(BSTNode<E>\* T)//height of tree

{

int leftH,rightH,hl;

if(T==0)

hl=0;

else

{

leftH=TreeDepth(T->left());

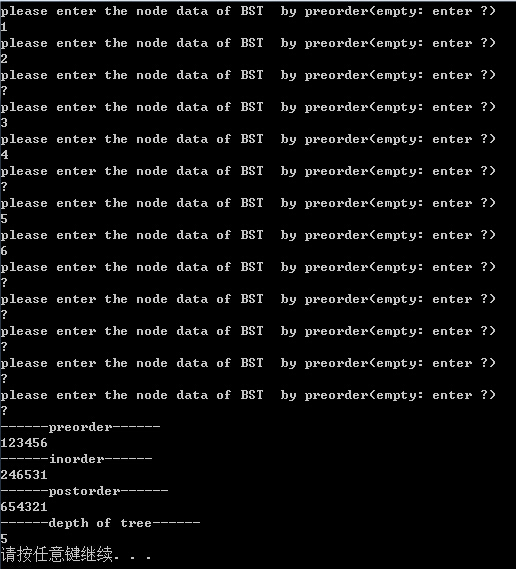
rightH=TreeDepth(T->right());

hl=(leftH>rightH?leftH:rightH)+1;

}

return hl;

}



(5.9)

int LeafNumber(BSTNode<E>\* T)//count leaf node

{

int number=0;

if(T!=NULL)

{

if(T->left()==0&&T->right()==0)

number++;

else

{

number=LeafNumber(T->left())+LeafNumber(T->right());

}

}

return number;

}

***BinaryTree.cpp***

#include"Tree.h"

#include<iostream>

using namespace std;

int main()

{

int x=1;

while(x==1)

{

Tree<char> bt; //create a tree

BSTNode<char>\* root = bt.Getroot( );

cout<<"------preorder------ "<<endl;

bt.preorder(root);

cout<<endl;

cout<<"------inorder------ "<<endl;

bt.inorder(root);

cout<<endl;

cout<<"------postorder------ "<<endl;

bt.postorder(root);

cout<<endl;

cout<<"------depth of tree------ "<<endl;

cout<<bt.TreeDepth(root)-1;

cout<<endl;

cout<<"------number of leaves node------ "<<endl;

cout<<bt.LeafNumber(root);

cout<<endl;

system("pause");

return 0;

}

