安 徽 新 华 学 院 实 验 报 告

姓 名 张卢 学号 1632104249 专业班级 16软件2班

课程名称 数据结构 实验名称 二叉树存储结构的实现及其应用

实验日期 2017.11.6 同组人员 指导老师 汪红霞 得分

**实 验 项 目 五**

**【实验名称】**二叉树存储结构的实现及其应用

**【实验类型】**基础性实验

**【实验目的与要求】**

* 掌握利用二叉链表方法建立二叉树；
* 掌握实现二叉树的前、中、后序三种遍历算法；
* 理解运用遍历算法实现二叉树的其他操作，如计算二叉树结点个数、叶子结点个数、二叉树的高度等；
* 理解运用递归方式描述算法及编写递归C/C++程序的方法，提高算法分析和程序设计能力。

**【实验主要仪器设备及台套数】**

一人一台安装Visual C++的电脑

**【实验学时】** 4学时

**【实验内容】**

1. 实现二叉树链式存储的各种操作（建立二叉树、遍历二叉树、计算二叉树结点个数、叶子结点个数、二叉树的高度等）的实现。

#include<iostream.h>

typedef char elemtype;

int jieDian=0;

class bitree

{ public:

elemtype data;

bitree \*lchild,\*rchild;

bitree \*creat();

void preorder(bitree \*root);

void preorder1(bitree \*root);

void inorder(bitree \*root);

void inorder1(bitree \*root);

void postorder(bitree \*root);

void postorder1(bitree \*root);

int LeafNodes(bitree \*b);

int depth(bitree \*root);

};

bitree \*bitree::creat()

{

bitree \*q[100]; //定义q数组作为队列，存放二叉链表中的结点，100为最大容量

bitree \*s; //二叉链表中的结点

bitree \*root ; //二叉链表的根指针

int front=1,rear=0; //定义队列的头、尾指针

char ch; //结点的data域值

root=NULL;

cin>>ch;

while(ch!='#') //输入值为#号，算法结束

{ s=NULL;

if(ch!=',') //输入数据不为逗号，表示不为虚结点，否则为虚结点

{ s=new bitree;

s->data=ch;

s->lchild=NULL;

s->rchild=NULL; }

rear++; q[rear]=s;

if(rear==1) root=s; else

{ if((s!=NULL)&&(q[front]!=NULL))

{ if(rear%2==0) q[front]->lchild=s;

else q[front]->rchild=s;}

if(rear%2==1) front++;

} cin>>ch;}

return root;

}

void bitree::preorder(bitree \*root)

{

bitree \*p;

p=root;

if(p!=NULL)

{cout<<p->data<<" ";

preorder(p->lchild);

preorder (p->rchild);

jieDian++;

}

}

void bitree::preorder1(bitree \*root)

{ bitree \*p,\*s[100];

int top=0;

p=root;

while((p!=NULL)||(top>0))

{ while(p!=NULL)

{cout<<p->data<<" ";

s[++top]=p;

p=p->lchild;

}

p=s[top--];

p=p->rchild;

}

}

void bitree::inorder(bitree \*root)

{ bitree \*p;

p=root;

if (p!=NULL)

{ inorder(p->lchild);

cout<<p->data<<" ";

inorder(p->rchild);

}

}

void bitree::inorder1( bitree \*root)

{ bitree \*p,\*s[100]; //s为一个栈，top为栈顶指针

int top=0;

p=root;

while((p!=NULL)||(top>0))

{

while(p!=NULL)

{

s[++top]=p; //进栈

p=p->lchild;} //进入左子树

{

p=s[top--]; //退栈

cout<<p->data<<" ";

p=p->rchild; //进入右子树

}

}

}

void bitree::postorder(bitree \*root)

{

bitree \*p;

p=root;

if(p!=NULL)

{

postorder(p->lchild);

postorder(p->rchild);

cout<<p->data<<" ";

}

}

void bitree::postorder1(bitree \*root)

{

bitree \*p,\*s1[100];

int s2[100],top=0,b;

p=root;

do

{

while(p!=NULL)

{

s1[top]=p;

s2[top++]=0;

p=p->lchild;

}

if(top>0)

{

b=s2[--top];

p=s1[top];

if(b==0)

{

s1[top]=p;

s2[top++]=1;

p=p->rchild;

}

else

{

cout<<p->data<<" ";

p=NULL;

}

}

}

while(top>0);

}

int bitree::LeafNodes(bitree \*b)

{

int num1,num2;

if(b==NULL)

return 0;

else if(b->lchild==NULL&&b->rchild==NULL)

return 1;

else

{

num1=LeafNodes(b->lchild);

num2=LeafNodes(b->rchild);

return (num1+num2);

}

}

int bitree::depth(bitree \*root)

{

if(root==NULL)

return 0;

else

{

int dep1=depth(root->lchild);

int dep2=depth(root->rchild);

if(dep1>dep2)

return dep1+1;

else

return dep2+1;

}

}

void main()

{

bitree b,\*res;

cout<<"创建二叉树: ";

res=b.creat();

cout<<"前根遍历的结果为: ";

b.preorder(res);

cout<<'\n';

cout<<"共有"<<jieDian<<"个结点"<<endl;

cout<<"叶子结点数为: "<<b.LeafNodes(res)<<endl;

cout<<"叶子深度为: "<<b.depth(res)<<endl;

cout<<"前根非递归遍历的结果为: ";

b.preorder1(res);

cout<<'\n';

cout<<"中根遍历的结果为: ";

b.inorder(res);

cout<<'\n';

cout<<"中根非递归遍历的结果为: ";

b.inorder1(res);

cout<<'\n';

cout<<"后根遍历的结果为: ";

b.postorder(res);

cout<<'\n';

cout<<"后根非递归遍历的结果为: ";

b.postorder1(res);

cout<<'\n';

}

**【实验心得】**

**深度体会并**掌握利用二叉链表方法建立二叉树以及实现二叉树的前、中、后序三种遍历算法；计算出二叉树结点个数、叶子结点个数、二叉树的高度等；理解运用递归方式描述算法及编写递归C/C++程序的方法，提高算法分析和程序设计能力,受益匪浅