

数据结构实验

报告

（2016/2017学年 第 一 学期）

题 目： 二叉树的基本运算和运用

**专 业** 电子信息工程

**学 号**  B15011209

**姓 名 王昆**

**指 导 教 师 李华康**

**日 期 2016 年 12月 14日**

**二叉树的基本运算和运用**

1. 实验目的
2. 理解二叉树的数据结构
3. 掌握二叉链表上实现二叉树基本运算的方法
4. 学会设计基于遍历的，求解二叉树应用问题的算法
5. 实习内容和要求
6. 实现二叉树的ADT的各基本运算。

（2）设计递归算法：

①删除一棵二叉树

②求一棵二叉树的高度

③求一棵二叉树中叶子结点个数

④复制一棵二叉树

⑤交换一棵二叉树

（3）设计算法，按自上而下，自左向右的次序，按层次遍历一棵二叉树。

（4）设计MAIN函数，测试上述每个运算。

1. 主函数流程图

构造二叉树

主函数

交换二叉树的左右子树

输出新的二叉树

复制二叉树

输出构造的二叉树

删除二叉树

输出新的二叉树

1.主函数代码

#include "stdio.h"

#include "stdlib.h"

#include "Btree.h"

int main()

{

BT \*T;

printf("请输入二叉树的根节点,并用0表示该节点为空\n");

T=CreatBT();

show(T);

printf("\n\nThis tree's high is: %d\n\n",height(T));

printf("\n\nThis tree's leaf is: %d\n\n",leaf(T));

CopyTree(T);

exchange(T);

show(T);

printf("\n");

DeleteBiTree(T);

printf("\n");

return 0;

}

2.功能函数

#include "stdio.h"

#include "stdlib.h"

#include "Btree.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*建立二叉树\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BT \*CreatBT()

{

BT \*t;

int x;

scanf("%d",&x);

//getchar();

if(x==0)

{

t=NULL;

}

else

{

t=(BT\*)malloc(sizeof(BT));

t->data=x;

printf("\n请输入%d结点的左子结点:",t->data );

t->lchild=CreatBT();

printf("\n请输入%d结点的右子结点:",t->data );

t->rchild=CreatBT();

}

return t;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\* //打印二叉树\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void show(BT \*T)

{

{

if(T==NULL) return;

else

{

printf("%3d",T->data);

show(T->lchild);

show(T->rchild);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*求取树的高度\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int height(BT \*T)

{

int lh,rh,h=0;

if(T==NULL)

h=0;

else

{

lh=height(T->lchild);

rh=height(T->rchild);

h=(lh>rh ? lh:rh)+1;

}

return h;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*删除一棵二叉树\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DeleteBiTree(BT\* root)

{

if(root==NULL)

return;

DeleteBiTree(root->lchild);

DeleteBiTree(root->rchild);

root=NULL;

return;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*求取二叉树中的叶子节点数\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int leaf(BT \*T)

{

if(T==NULL)

return 0;

else

if(T->lchild==NULL && T->rchild==NULL)

return 1;

else

return leaf(T->lchild)+leaf(T->rchild);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*复制一棵二叉树\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BT \*CopyTree(BT \*root)

{

BT \* newnode;

if ( root == NULL )

return NULL;

else

{

newnode = ( BT\* ) malloc(sizeof(BT));

newnode->data = root->data;

newnode->lchild = CopyTree(root->lchild);

newnode->rchild = CopyTree(root->rchild);

return newnode;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*交换二叉树的左右子树\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void exchange(BT \*rt)

{

BT \*temp = NULL;

if(rt->lchild == NULL && rt->rchild == NULL)

return;

else{

temp = rt->lchild;

rt->lchild = rt->rchild;

rt->rchild = temp;

}

if(rt->lchild)

exchange(rt->lchild);

if(rt->rchild)

exchange(rt->rchild);

}

此部分主要完成了各个功能函数的c语言代码实现

3.功能函数的头函数

#include "stdio.h"

#include "stdlib.h"

struct BT

{

int data;

struct BT \*lchild,\*rchild;

};

typedef struct BT BT;

BT \*CreatBT( );

void show(BT \*T);

int height(BT \*T);

int leaf(BT \*T);

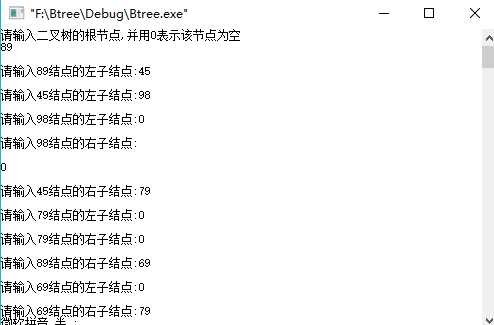
BT \*CopyTree(BT \*root);

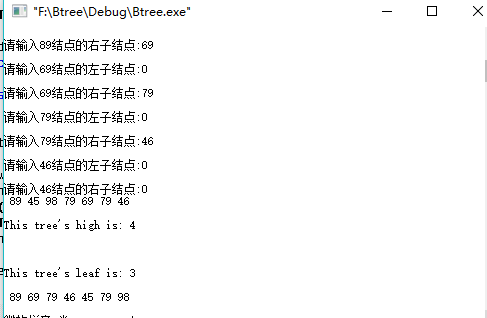
void exchange(BT \*rt);

void DeleteBiTree(BT\* root);

此部分主要定义了各个函数，以及二叉树

1. 程序调试





1. 实验心得

这次实验掌握了对链表的基本操作，熟悉了二叉树在c语言环境下的操作方法，基本掌握了二叉树的一些基本操作，掌握了工程的建立方法以及调试，同时也掌握了在编译出现问题的时候快速的处理问题。

1. 问题及解决方法

删除二叉树的过程中，在输出结果以观察是否正确时，发现二叉树删除失败

解决办法：在重读程序思考之后发现，由于二叉树已经经历过复制所以在删除原而擦函数过后，复制之后的二叉树依旧保留，所以在删除过原二叉树之后，打印的是复制过后的二叉树