实验4二叉树实验

4.1实验目的

1. 掌握二叉树的动态链表存储结构及表示。
2. 掌握二叉树的三种遍历算法（递归和非递归两类）。
3. 运用二叉树三种遍历的方法求解有关问题。

4.2 实验要求

1. 结构定义和算法实现放入库文件，如“BiTree.h”中；
2. 二叉树的测试数据用文本文件方式给出，例如测试数据名为bt151.btr的二叉树，可参考发来的二叉树形状和参考存储文件；
3. 二叉树创建方法可自行选择；
4. 可多次连续测试。

4.3 实验任务

设计二叉树的二叉链表存储结构，编写算法实现下列问题的求解。

1. 设计算法按中序次序输出二叉树中各结点的值及其所对应的层次数。

实验测试数据基本要求：

第一组数据： bt151.btr

第二组数据： bt21.btr

1. 求二叉树的叶子结点数和1度结点数。

实验测试数据基本要求：

第一组数据： bt261.btr

第二组数据： bt21.btr

1. 键盘输入一个元素x，求其父节点、兄弟结点、子结点的值，不存在时给出相应提示信息。对兄弟结点和孩子结点，存在时要明确指出是左兄弟、左孩子、右兄弟或右孩子。

实验测试数据基本要求：

第一组数据： bt31.btr

第二组数据： bt21.btr

1. 键盘输入一个元素x，求其在树中的层次。不存在时给出相应提示信息。

实验测试数据基本要求：

第一组数据： bt26.btr

第二组数据： bt21.btr

1. 将按顺序方式存储在数组中的二叉树转换为二叉链表形式。（数组中要扩展为完全二叉树）。

实验测试数据基本要求：

第一组数据： bt8.btr

第二组数据： bt14.btr

1. 输出二叉树从每个叶子结点到根结点的路径（经历的结点）。

实验测试数据基本要求：

第一组数据： bt261.btr

第二组数据： bt21.btr

1. 对二叉链表表示的二叉树，按从上到下，从左到右打印结点值，即按层次遍历序打印。（提示：需要使用队列）

实验测试数据基本要求：

第一组数据： bt261.btr

第二组数据： bt21.btr

4.4\* 二叉树扩展实验

非必做内容，有兴趣的同学选做，

1. 复制一棵二叉树T到T1。

实验测试数据基本要求：

第一组数据： bt151.btr

第二组数据： bt21.btr

1. 交换二叉树中每个结点的左右孩子指针的值。（即：左子树变为右子树，右子树变为左子树）。

实验测试数据基本要求：

第一组数据： bt151.btr

第二组数据： bt21.btr

1. 对二叉链表表示的二叉树，求2个结点最近的共同祖先。

实验测试数据基本要求：

第一组数据： bt261.btr

第二组数据： bt21.btr

4.5 二叉树二叉链表存储创建和销毁参考

**1. 数据文件输入创建二叉树**

推荐一种基于文本文件的二叉树创建方法，即将二叉树的结点信息保存在一个文本文件中，然后用程序自动读入来创建二叉树。我们先来定义文本文件的格式。

标识行：BinaryTree—标识这是一个二叉树的数据文件。

结点行：每个结点一行，结点严格按照先序遍历次序排列。每行3列。第1列为结点数据；第2列标识有无左子树，1—有左子树，0—无左子树；第3列标识有无右子树，1—有右子树，0—无右子树。

a

b

c

图一棵二叉树

g

i

d

e

f

h

T

对如图所示二叉树，完整数据文件如下：

BinaryTree

a 1 1

b 1 1

d 0 0

e 1 1

g 0 0

h 0 0

c 1 0

f 0 1

i 0 0

数据文件的扩展名随意，只要按文本文件读写即可，例如上述文件不妨命名为BiTree9.bit。

接下来介绍基于上述数据文件创建二叉树的函数。

【结点数据读入数组函数】

因为是按先序次序递归创建二叉树，在创建过程中又要记录读取数据的行数，直接从文件中读取数据创建时处理不方便，所以我们先把文件中的数据读取到一个二维数组中，再从数组中读取数据来创建二叉树。从文件读取数据到数组的函数代码如下：

bool ReadFileToArray(char fileName[], char strLine[100][3], int & nArrLen)

{ // fileName[]存放文件名

// strLine[][3]存放结点的二维数组，数组的3列对应数据文件的3列

// nArrLen返回二叉树结点的个数

FILE\* pFile; //定义二叉树的文件指针

char str[1000]; //存放读出一行文本的字符串

pFile=fopen("BiTree9.bit","r");

if(!pFile)

{

printf("二叉树数据文件打开失败！\n");

return false;

}

//读取文件第1行，判断二叉树标识BinaryTree是否正确

if(fgets(str,1000,pFile)!=NULL)

{

if(strcmp(str,"BinaryTree\n")!=0)

{

printf("打开的文件格式错误！\n");

fclose(pFile); //关闭文件

return false;

}

}

nArrLen=0;

while(fscanf(pFile,"%c %c %c\n",

&strLine[nArrLen][0],&strLine[nArrLen][1],&strLine[nArrLen][2])!=EOF)

{ //循环读取结点行数据，存入数组，结点总数加1

nArrLen++;

}

fclose(pFile); //关闭文件

return true;

}

【从数组数据按先序次序创建二叉树】

bool CreateBiTreeFromFile(BiNode\* & pBT, char strLine[100][3],int nLen, int & nRow)

{

//strLine[100][3]--保存结点数据的二维数组

//nLen--结点个数

//nRow--数组当前行号

if((nRow>=nLen) || (nLen==0))

return false; //数据已经处理完毕，或者没有数据，退出

//根据数组数据递归创建二叉树

pBT=new BiNode; //建立根结点

pBT->data=strLine[nRow][0];

pBT->lChild=NULL;

pBT->rChild=NULL;

int nRowNext=nRow; //保留本次递归的行号

if(strLine[nRowNext][1]=='1')

{ //当前结点有左子树，读下一行数据，递归调用创建左子树。

nRow++; //行号加1

CreateBiTreeFromFile(pBT->lChild, strLine,nLen,nRow);

}

if(strLine[nRowNext][2]=='1')

{ //当前结点有右子树，读下一行数数，递归调用创建右子树。

nRow++; //行号加1

CreateBiTreeFromFile(pBT->rChild, strLine,nLen,nRow);

}

return true;

}

**2. 二叉树的销毁**

创建了二叉链表存储结构的二叉树，使用完毕后应当释放此二叉树占用的内存，因为在创建二叉树时使用malloc()函数或new操作符动态申请了内存，当这个二叉树不再需要时，必须手工释放动态申请的内存，否则造成内存泄漏。下面给出销毁二叉链表存储结构二叉树的程序。

void DestroyBiTree(BiNode\* pBT)

{

if(pBT)

{

DestroyBiTree(pBT->lChild); //递归销毁左子树

DestroyBiTree(pBT->rChild); //递归销毁右子树

delete pBT; //释放当前根结点

}

}