数据结构实验报告

评分

满分——5分

学号：2015111833 姓名：邓金红 专业：计算机科学与技术

知识范畴：树 完成日期：2016年04月28日

实验题目：顺序存储完全二叉树先、中、后序遍历

实验内容及要求：

输入一个字符串，存储于一维数组。以该一维数组作为完全二叉树的存储结构，实现先、中、后序遍历，输出遍历结果。

将该完全二叉树转换为二叉链表存储结构，然后基于二叉链表存储结构再次进行先、中、后序遍历并输出遍历结果。

实验目的：掌握完全二叉树的顺序存储与链式存储结构以及遍历算法。

数据结构设计简要描述：

采用一维数组的形式按层序遍历的顺序存储完全二叉树，然后通过转换函数将其转换为二叉链表的存储结构。

算法设计简要描述：

根据分析，完全二叉树的层序次序存储有如下性质(假设下标从0开始)：

序号i的左儿子的下标为2\*i+1， 右儿子的下标为2\*i+2.

由此按照递归思想，只需要每次递归时判断儿子的下标是否在合理范围内。若在，按照前、中、后序遍历的思想遍历即可。

输入/输出设计简要描述：

输入：字符串文件读入

输出：1、一维数组作为完全二叉树的存储结构，先、中、后序遍历，输出遍历结果。

2、二叉链表存储结构再次进行先、中、后序遍历并输出遍历结果。

编程语言说明：

使用Visual C++编程。 主要代码采用C语言实现 ；动态存储分配采用C++的new和delete操作符实现；输入与输出采用C++的cin和cout流；程序注释采用C/C++规范。

主要函数说明：

void preOrder(char \*s, int root)//一维数组前序遍历

void inOrder(char \*s, int root) //一维数组中序遍历

void lasOrder(char \*s, int root) //一维数组后序遍历

void array\_to\_bitree(BiTree \*root, char \*a, int n)//一维数组转二叉链表存储结构

void bi\_preOrder(BiTree root)// 二叉链表存储结构前序遍历

void bi\_inOrder(BiTree root) // 二叉链表存储结构中序遍历

void bi\_lasOrder(BiTree root) // 二叉链表存储结构后序遍历

程序测试简要报告：

***测试样例一：***

输入：ABCDEFG

输出：

一维数组遍历结果：

前序遍历：ABDECFG

中序遍历：DBEAFCG

后序遍历：DEBFGCA

一维数组转换二叉链表遍历结果：

前序遍历：ABDECFG

中序遍历：DBEAFCG

后序遍历：DEBFGCA

***结论***

程序输出结果与期望输出结果相符。

***测试样例二：***

输入：ABCDEFGHIJ

输出：

一维数组遍历结果：

前序遍历：ABDHIEJCFG

中序遍历：HDIBJEAFCG

后序遍历：HIDJEBFGCA

一维数组转换二叉链表遍历结果：

前序遍历：ABDHIEJCFG

中序遍历：HDIBJEAFCG

后序遍历：HIDJEBFGCA

***结论：***

程序输出结果与期望输出结果相符。

***测试样例三：***

输入：ABCDEFGHIJKLMN

输出：一维数组遍历结果：

前序遍历：ABDHIEJKCFLMGN

中序遍历：HDIBJEKALFMCNG

后序遍历：HIDJKEBLMFNGCA

一维数组转换二叉链表遍历结果：

前序遍历：ABDHIEJKCFLMGN

中序遍历：HDIBJEKALFMCNG

后序遍历：HIDJKEBLMFNGCA

***结论：***

程序输出结果与期望输出结果相符。

源程序代码：

#include <iostream>

#include <fstream>

#include <string.h>

using namespace std;

char s[100]; //输入字符串

typedef struct node

{

char data;

struct node \*lchild, \*rchild;

}\*BiTree, BiNode;

void preOrder(char \*s, int root) //一维数组前序遍历

{

if(s[root])

{

cout << s[root];

if(s[root\*2+1] != '\0')

preOrder(s, root\*2+1);

if(s[root\*2+2] != '\0')

preOrder(s, root\*2+2);

}

}

void inOrder(char \*s, int root) //一维数组中序遍历

{

if(s[root])

{

if(s[root\*2+1] != '\0')

inOrder(s, root\*2+1);

cout << s[root];

if(s[root\*2+2] != '\0')

inOrder(s, root\*2+2);

}

}

void lasOrder(char \*s, int root) //一维数组后序遍历

{

if(s[root])

{

if(s[root\*2+1] != '\0')

lasOrder(s, root\*2+1);

if(s[root\*2+2] != '\0')

lasOrder(s, root\*2+2);

cout << s[root];

}

}

void array\_to\_bitree(BiTree \*root, char \*a, int n) //一维数组转二叉链表存储结构

{

for(int i = 0; i < n; i++)

root[i] = new BiNode();

for(int i = 0; i <= n-1; i++)

{

root[i]->data = a[i];

if(i\*2 + 1 <= n-1)

root[i]->lchild = root[i\*2 + 1];

else

root[i]->lchild = root[i]->rchild = NULL;

if(i\*2 + 2 <= n-1)

root[i]->rchild = root[i\*2 + 2];

else

root[i]->rchild = NULL;

}

}

void bi\_preOrder(BiTree root) //二叉链表存储结构前序遍历

{

if(root)

{

cout << root->data;

bi\_preOrder(root->lchild);

bi\_preOrder(root->rchild);

}

}

void bi\_inOrder(BiTree root) //二叉链表存储结构中序遍历

{

if(root)

{

bi\_inOrder(root->lchild);

cout << root->data;

bi\_inOrder(root->rchild);

}

}

void bi\_lasOrder(BiTree root) //二叉链表存储结构后序遍历

{

if(root)

{

bi\_lasOrder(root->lchild);

bi\_lasOrder(root->rchild);

cout << root->data;

}

}

int main()

{

ifstream cin("input.txt");

cin >> s;

//一位数组直接遍历

cout << "一位数组遍历结果：" << endl;

cout << "前序遍历：";

preOrder(s, 0);cout << endl;

cout << "中序遍历：";

inOrder(s, 0);cout << endl;

cout << "后序遍历：";

lasOrder(s, 0);cout << endl;

cout << "一位数组转换二叉链表遍历结果：" << endl;

BiTree bt[100];//二叉链表存储结构

array\_to\_bitree(bt, s, strlen(s));

cout << "前序遍历：";

bi\_preOrder(\*bt);cout << endl;

cout << "中序遍历：";

bi\_inOrder(\*bt);cout << endl;

cout << "后序遍历：";

bi\_lasOrder(\*bt);cout << endl;

return 0;

}