

天津商业大学学生实验报告

开课实验室：信息实验室

开课时间 13 年 9 月 1 日

实验报告 13 年 10 月 31 日

学院名称	信息工程	年级、专业、班	软件工程 12-01 班	学号	20125041	姓名	王靖伟	同组姓名	
课程名称	数据结构	实验项目名称	实验三 二叉树			指导教师	黄橡丽		
实验类型	验证 <input checked="" type="checkbox"/> 综合 <input type="checkbox"/> 设计 <input type="checkbox"/> 创新 <input type="checkbox"/>						成绩		
教师评语	教师签名：_____ 年 月 日								
实验报告内容一般包括以下几个内容：1、目的要求 2、仪器用具及材料（仪器名称及主要规格、用具名称） 3、实验内容及原理（简单但要抓住要点，写出依据原理） 4、操作方法与实验步骤 5、数据图表格（照片） 6、实验过程原始记录 7 数据处理及结果（按实验要求处理数据、结论） 8、作业题 9、讨论（对实验中存在的问题、进一步的想法等进行讨论）									
实验报告内容： 1. 实验目的、要求： (1) 进一步掌握树的结构及非线性特点，递归特点和动态性。 (2) 掌握二叉树的建立算法。 (3) 掌握二叉树的三种遍历方法以及基于遍历的几种基本操作。 2. 实验内容： (1) 实验题目 1. 二叉树的链式存储结构的建立。 2. 二叉树的三种遍历算法以及基于遍历的几种操作的实现。 3. 求二叉树的叶子结点数。 4. 求二叉树的深度。 (2) 算法设计思想（说明整个程序由一个主函数和哪几个函数组成，并给出主要函数的算法设计思想） 1. 建立二叉树存储结构 <code>Status CreateBiTree(BiTree &T)</code> 2. 先序遍历二叉树 <code>void PreOrder(BiTree T)</code> //若二叉树为空，则空操作，否则 a.访问根结点,b.先序遍历左子树,c.先序遍历右子树。 3. 中序遍历二叉树 <code>void InOrder(BiTree T)</code> //若二叉树为空，则空操作，否则 a.先序遍历左子树,b.访问根结点,c.先序遍历右子树。 4. 后序遍历二叉树 <code>void PostOrder(BiTree T)</code> //若二叉树为空，则空操作，否则 a.先序遍历左子树,b.先序遍历右子树,c.访问根结点。 5. 求二叉树的叶子结点数 <code>void CountLeaf(BiTree T,int& count)</code> //若一个结点既没有左孩子又没有右孩子，计数器自加 1，否则先求左子树的叶子结点数， //后求右子树的叶子结点数。 6. 求二叉树中度为 1 的结点和度为 2 的结点的数 <code>void CountBT(BiTree T,int &m,int &n)</code> 7. 求二叉树的深度 <code>int Depth(BiTree T)</code>									

8.主函数 void main()

(3) 程序清单

行号	代码
1	#include "iostream.h"
2	#include "stdio.h"
3	#include "stdlib.h"
4	#define OK 1
5	#define ERROR 0
6	#define OVERFLOW -2
7	#define NULL 0
8	typedef char TElemType;
9	typedef int Status;
10	typedef struct BiTNode { // 结点结构
11	TElemType data;
12	struct BiTNode *lchild, *rchild; // 左右孩子指针
13	} BiTNode, *BiTree;
14	//以下是建立二叉树存储结构
15	Status CreateBiTree(BiTree &T) {
16	char ch;
17	scanf("%c",&ch);
18	if (ch=='#') T = NULL;
19	else {
20	if (!(T = (BiTNode *)malloc(sizeof(BiTNode))))
21	exit(OVERFLOW);
22	T->data = ch; // 生成根结点
23	CreateBiTree(T->lchild); // 构造左子树
24	CreateBiTree(T->rchild); // 构造右子树
25	}
26	return OK;
27	} // CreateBiTree
28	void PreOrder (BiTree T)
29	{ // 先序遍历二叉树
30	if (T) {
31	printf("%4c",T->data); // 访问结点
32	PreOrder(T->lchild); // 遍历左子树
33	PreOrder(T->rchild); // 遍历右子树
34	}
35	}
36	void InOrder (BiTree T)
37	{ // 中序遍历二叉树
38	//请将该算法补充完整
39	if (T) {
40	InOrder(T->lchild); // 遍历左子树
41	printf("%4c",T->data); // 访问结点
42	InOrder(T->rchild); // 遍历右子树
43	}
44	}

45	void PostOrder (BiTree T)
46	{ // 后序遍历二叉树
47	//将该算法补充完整
48	if (T) {
49	PostOrder(T->lchild); // 遍历左子树
50	PostOrder(T->rchild); // 遍历右子树
51	printf("%4c",T->data); // 访问结点
52	}
53	}
54	//求二叉树的叶子结点数
55	void CountLeaf (BiTree T, int& count){
56	//将该算法补充完整
57	if(T){
58	if((!T->lchild)&&(!T->rchild))
59	count++;
60	CountLeaf(T->lchild,count);
61	CountLeaf(T->rchild,count);
62	}
63	}
64	//求二叉树中度为 1 的结点和度为 2 的结点的数
65	void CountBT(BiTree T,int &m,int &n)
66	{
67	if(T){
68	if((T->lchild!=0)&&(T->rchild!=0))
69	n++; //度为 2 的结点
70	else if(((T->lchild!=0)&&(T->rchild==0)) ((T->lchild==0)&&(T->rchild!=0)))
71	m++; //度为 1 的结点
72	CountBT (T->lchild,m,n);
73	CountBT (T->rchild,m,n);
74	}
75	}
76	//以下是求二叉树的深度
77	int Depth (BiTree T){
78	int m,n;
79	if(!T) return 0;
80	else{
81	m = Depth(T->lchild);
82	n = Depth(T->rchild);
83	return (m>n?m:n)+1;
84	}
85	}
86	void main(){
87	BiTree T;
88	int s=0,m=0,n=0,d=0;
89	T=NULL;
90	int select;
91	while(1){
92	printf("\n 请选择要执行的操作:\n");

93	printf("1.创建二叉树\n");
94	printf("2.二叉树的递归遍历算法（前、中、后）\n");
95	printf("3.求二叉树的叶子结点数\n");
96	printf("4.求二叉树的深度\n");
97	printf("0.退出\n");
98	scanf("%d",&select);
99	getchar();
100	switch(select){
101	case 0:return;
102	case 1:
103	printf("\n 请按先序次序输入各结点的值，以#表示空树:\n");
104	CreateBiTree(T);
105	printf("二叉树已建立完毕！ \n");
106	break;
107	case 2:
108	if(!T) printf("\n 未建立树，请先建树！ \n");
109	else{
110	printf("\n 先序遍历:");
111	PreOrder(T);
112	printf("\n 中序遍历:");
113	InOrder(T);
114	printf("\n 后序遍历:");
115	PostOrder(T);
116	printf("\n");
117	}
118	break;
119	case 3:
120	if(!T) printf("\n 未建立树，请先建树！ \n");
121	else{
122	CountLeaf(T,s);
123	printf("\n 叶子结点数为: %d\n",s);
124	CountBT(T,m,n);
125	printf("度为 1 的结点数为: %d\n",m);
126	printf("度为 2 的结点数为: %d\n",n);
127	}
128	break;
129	case 4:
130	if(!T) printf("\n 未建立树，请先建树！ \n");
131	else{
132	d=Depth(T);
133	printf("\n 二叉树的深度为: %d\n",d);
134	}
135	break;
136	default:
137	printf("请确认选择项:\n");
138	}//end switch
139	}//end while
140	}

(3) 测试数据

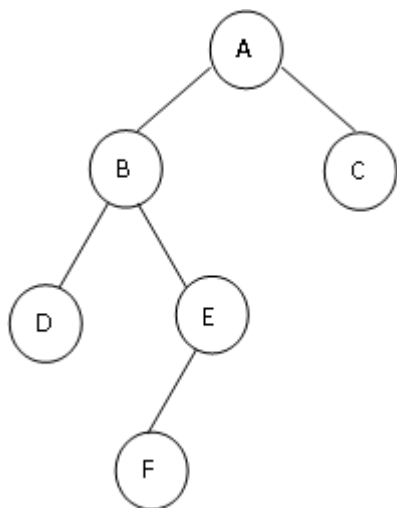


图 1.测试用例 1——ABD##EF###C##

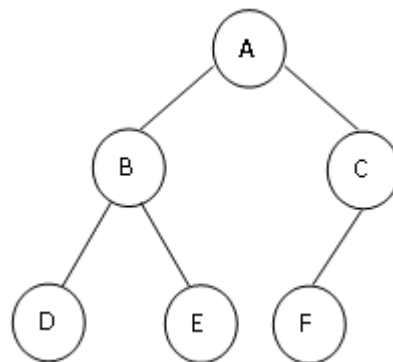


图 2.测试用例 2——ABD##E##CF###

(4) 实验结果

```

F:\★总表\课程\数据结构\实验\2013年10月31日 实验三\Debug\3.exe
请选择要执行的操作：
1.创建二叉树
2.二叉树的递归遍历算法（前、中、后）
3.求二叉树的叶子结点数
4.求二叉树的深度
0.退出
1
请按先序次序输入各结点的值，以#表示空树：
ABD##EF###C##
二叉树已建立完毕！

请选择要执行的操作：
1.创建二叉树
2.二叉树的递归遍历算法（前、中、后）
3.求二叉树的叶子结点数
4.求二叉树的深度
0.退出
2
先序遍历：  A   B   D   E   F   C
中序遍历：  D   B   F   E   A   C
后序遍历：  D   F   E   B   C   A
  
```

图 3.测试用例 1 的实验结果片段 a

```

F:\★总表\课程\数据结构\实验\2013年10月31日 实验三\Debug\3.exe
请选择要执行的操作：
1.创建二叉树
2.二叉树的递归遍历算法（前、中、后）
3.求二叉树的叶子结点数
4.求二叉树的深度
0.退出
3
叶子结点数为：3
度为1的结点数为：1
度为2的结点数为：2
  
```

图 4.测试用例 1 的实验结果片段 b

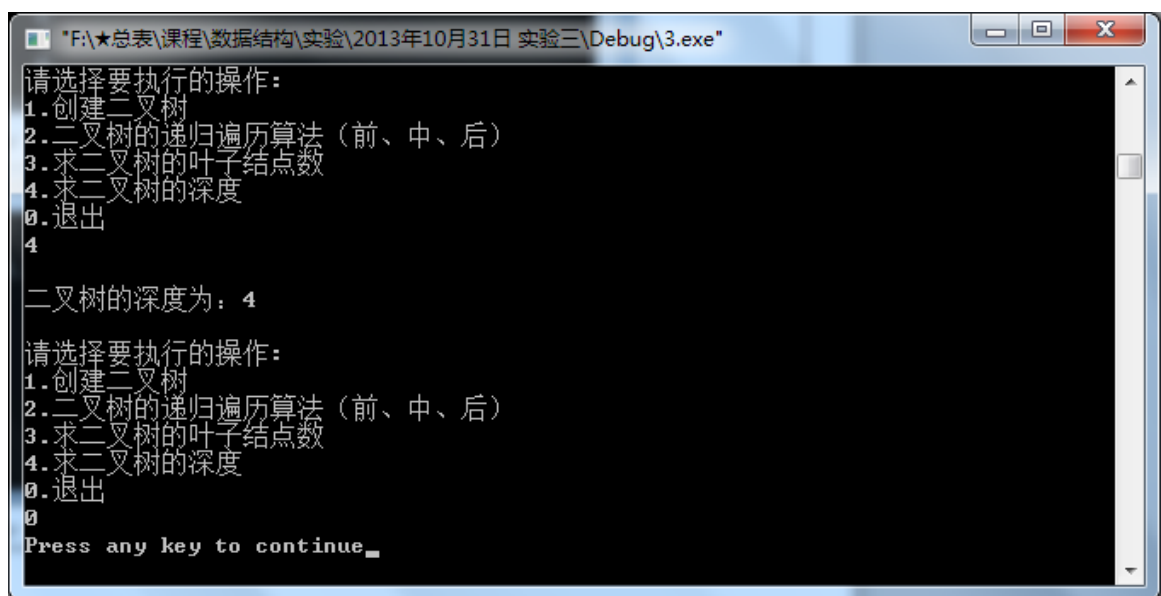


图 5.测试用例 1 的实验结果片段 c

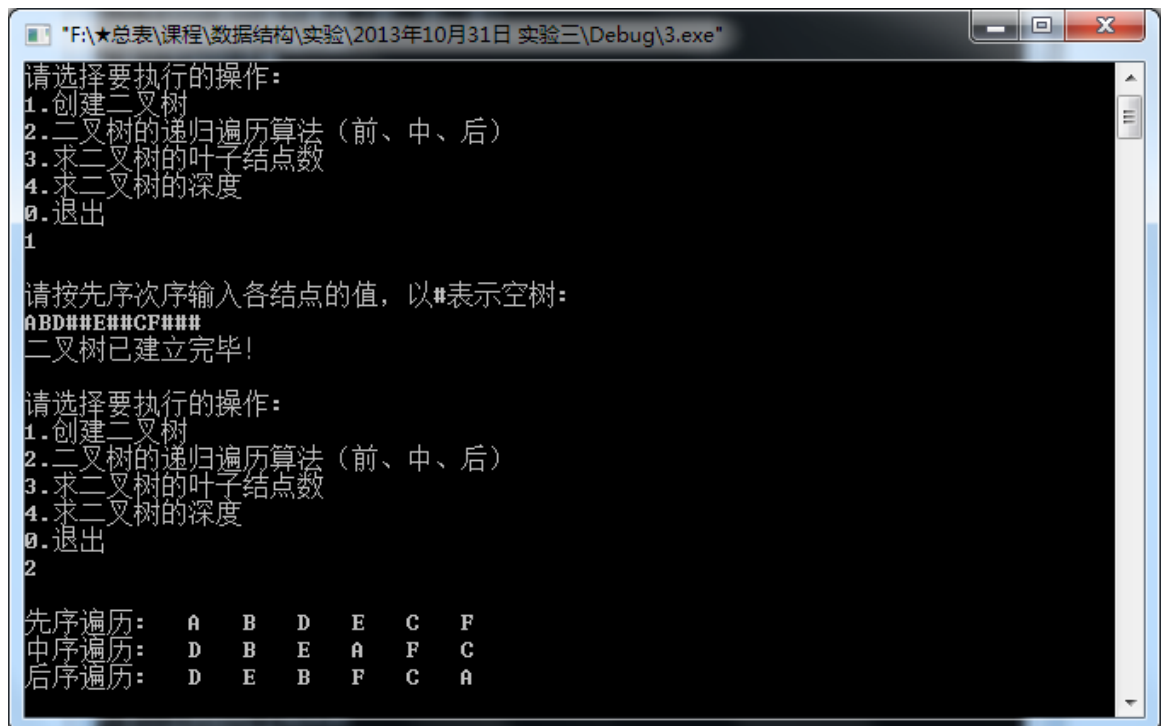


图 6.测试用例 2 的实验结果片段 a

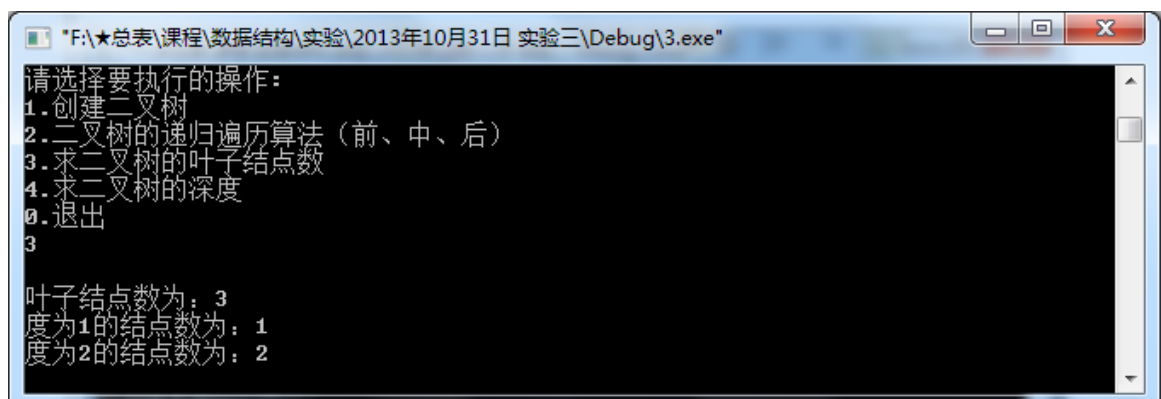
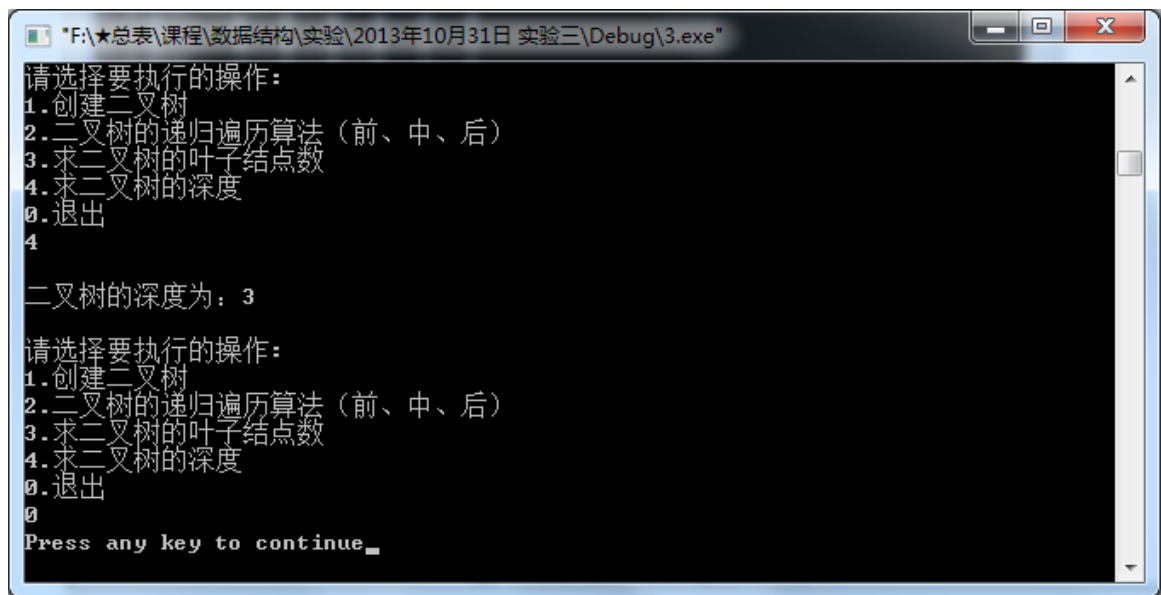


图 7.测试用例 2 的实验结果片段 b



```
F:\★总表\课程\数据结构\实验\2013年10月31日 实验三\Debug\3.exe
请选择要执行的操作:
1.创建二叉树
2.二叉树的递归遍历算法(前、中、后)
3.求二叉树的叶子结点数
4.求二叉树的深度
0.退出
4

二叉树的深度为: 3

请选择要执行的操作:
1.创建二叉树
2.二叉树的递归遍历算法(前、中、后)
3.求二叉树的叶子结点数
4.求二叉树的深度
0.退出
0
Press any key to continue_
```

图 8.测试用例 2 的实验结果片段 c

(5) 实验收获

通过本次上机实验,使我进一步掌握了树的结构及非线性特点、递归特点和动态性,以及二叉树的建立算法和二叉树的三种遍历方法以及基于遍历的几种基本操作。

注 1. 每个实验项目一份实验报告。2. 实验报告第一页学生必须使用规定的实验报告纸书写, 附页用实验报告附页纸或 A4 纸书写, 字迹工整, 曲线要画在坐标纸上, 线路图要整齐、清楚(不得徒手画)。3. 实验教师必须对每份实验报告进行批改, 用红笔指出实验报告中的错、漏之处, 并给出评语、成绩, 签全名、注明日期。 4. 待实验课程结束以后, 要求学生把实验报告整理好, 交给实验指导教师, 加上实验课学生考勤及成绩登记表(见附件 2)、目录和学院统一的封面(见附件 3)后, 统一装订成册存档。

制表单位: 设备处