**理工学院计算机类课程实验报告**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 课程名称 | 数据结构 | | 实验名称 | 实验5 Project 二叉树 |
| 小组名称及成员 | 吴警 | | | |
| 专 业 | 软件工程 | | 年 级 | 2019 |
| 班 级 |  | | 实验时间 | 第8周 |
| 教师评语（或成绩） | | 教师签字：  年 月 日 | | |
| **一、实验目的**  （1）了解C项目编写方式  （2）熟悉二叉树的基本运算以及算法与程序之间的关系 | | | | |
| **二、实验设备（工具、材料、硬软件）及要求**  1、设备：计算机一台、Dev C++  2、具体要求：  实现二叉树的基本算法：  （1）创建二叉树  （2）二叉树的遍历  （3）求二叉树的深度  （4）求二叉树的叶子数  **上述，各小组可根据实际情况，使用顺序存储或链式存储实现树的创建，在具体算法实现时可考虑递归和非递归两种写法。** | | | | |
| 三、实验过程  要求：（1）列出实现的源代码、运行效果截图；（2）为源码添加必要的注释语句。另外，对实验进一步分析存在的问题提出解决办法，或总结实验所取得的经验均可写入报告中。能力强者也可使用java语言实现代码。  源代码：  **main**.c  **#include <stdio.h>**  **#include <stdlib.h>**  **#include "bitree.h"**  **int main(int argc, char \*argv[]) {**  **BiTree BT; //创建二叉树**  **BT=createBiTree(BT); //键入二叉树**  **printf("\n");**  **firstTra(BT); //先序输出二叉树**    **int depth=0; //树的深度**  **depth=depthBiTree(BT); //调用函数求树的深度**  **printf("\n\ndepth=%d\n",depth);**    **int leaf=0; //树的叶子数**  **leaf=leafBiTree(BT); //调用函数求叶子数**  **printf("\nleaf=%d\n",leaf);**  **return 0;**  **}**  **二叉树结构体及其函数：**  **bitree.h**  **#include<stdio.h>**  **#include<malloc.h>**  **typedef int ElemType;**  **struct TreeNode{**  **ElemType data; //树的结点**  **struct TreeNode \*lift; //树的左子树**  **struct TreeNode \*right; //树的右子树**  **};**  **typedef struct TreeNode TNode;**  **typedef struct TreeNode \*BiTree;**  **//先序遍历 输出**  **void firstTra(BiTree BT){**  **if(BT){ //判断BT 不为空**  **printf("%d",BT->data); //输出根**  **firstTra(BT->lift); //递归遍历 左子树**  **firstTra(BT->right); //递归遍历 右子树**  **}**  **}**  **//中序遍历 输出**  **void midTra(BiTree BT){**  **if(BT){ //判断BT 不为空**  **midTra(BT->lift); //递归遍历 左子树**  **printf("%d",BT->data); //输出根**  **midTra(BT->right); //递归遍历 右子树**  **}**  **}**  **//后序遍历 输出**  **void endTra(BiTree BT){**  **if(BT){ //判断BT 不为空**  **endTra(BT->lift); //递归遍历 左子树**  **endTra(BT->right); //递归遍历 右子树**  **printf("%d",BT->data); //输出根**  **}**  **}**  **//创建二叉树，利用先序遍历的思想**  **BiTree createBiTree(BiTree BT){**  **int x;**  **scanf("%d",&x);**  **if(x==0) //判断x是否为空**  **BT=NULL;**  **else{**  **BT=(BiTree)malloc(sizeof(TNode)); //为新结点申请空间**  **BT->data=x; //创建根结点**  **BT->lift=createBiTree(BT->lift); //递归创建左子树**  **BT->right=createBiTree(BT->right); //递归创建右子树**  **}**  **}**  **//求二叉树深度，递归遍历**  **int depthBiTree(BiTree BT){**  **int l=0,r=0; // l表示左子树深度，r表示右子树深度**  **if(BT==NULL) //判断根是否为空**  **return 0;**  **else{**  **l=depthBiTree(BT->lift); //递归计算左子树的深度**  **r=depthBiTree(BT->right); //递归计算右子树的深度**  **}**  **if(l>r) //判断左右子树，返回数值较大的子树深度+1**  **return l+1;**  **else**  **return r+1;**  **}**  **//求二叉树的叶子数，遍历**  **int leafBiTree(BiTree BT){**  **int l=0,r=0; // l表示左子树叶子，r表示右子树叶子**  **if(BT==NULL) //判断根是否为空**  **return 0;**  **else if((BT->lift==NULL)&&(BT->right==NULL)){**  **return 1; //判断左右子树是否为空，若都为空返回1**  **}**  **else{**  **l=leafBiTree(BT->lift); //递归调用计算左子树叶子数**  **r=leafBiTree(BT->right); //递归调用计算右子树叶子数**  **}**  **return l+r; //返回左右叶子数之和**  **}**  运行结果截图：    实验过程中遇到的问题：  ·对递归调用计算左右子树深度的增加不太理解（l和r的增加）。    改进措施：  ·及时与组内成员交流问题。  ·加深对递归算法的学习。  收获：  ·学会了二叉树的创建及遍历算法。  ·巩固提升了对递归算法的了解与使用。  ·会利用算法计算二叉树的深度及其子叶数。 | | | | |