**Project 6 二叉树的遍历**

学号 2014211116 姓名 骆金参 上交时间 2015-11-15

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 题号 | 程序逻辑(40) | 算法新颖性(20) | 代码规范 (20) | 实验报告(20) | 总分 |
| 得分 |  |  |  |  |  |

1. **问题描述**

二叉树的创建，计算深度，以及前序中序后序遍历输出。

**2. 算法描述**

**#include "string.h"**

**#include "stdio.h"**

**#include "stdlib.h"**

**#include "io.h"**

**#include "math.h"**

**#include "time.h"**

**#define OK 1**

**#define ERROR 0**

**#define TRUE 1**

**#define FALSE 0**

**#define MAXSIZE 100 /\* 存储空间初始分配量 \*/**

**typedef int Status; /\* Status是函数的类型,其值是函数结果状态代码，如OK等 \*/**

**/\* 用于构造二叉树\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/**

**int index=1;**

**typedef char String[24]; /\* 0号单元存放串的长度 \*/**

**String str;**

**Status StrAssign(String T,char \*chars)**

**{**

**int i;**

**if(strlen(chars)>MAXSIZE)**

**return ERROR;**

**else**

**{**

**T[0]=strlen(chars);**

**for(i=1;i<=T[0];i++)**

**T[i]=\*(chars+i-1);**

**return OK;**

**}**

**}**

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

**typedef char TElemType;**

**TElemType Nil=' '; /\* 字符型以空格符为空 \*/**

**Status visit(TElemType e)**

**{**

**printf("%c ",e);**

**return OK;**

**}**

**typedef struct BiTNode /\* 结点结构 \*/**

**{**

**TElemType data; /\* 结点数据 \*/**

**struct BiTNode \*lchild,\*rchild; /\* 左右孩子指针 \*/**

**}BiTNode,\*BiTree;**

**/\* 构造空二叉树T \*/**

**Status InitBiTree(BiTree \*T)**

**{**

**\*T=NULL;**

**return OK;**

**}**

**/\* 初始条件: 二叉树T存在。操作结果: 销毁二叉树T \*/**

**void DestroyBiTree(BiTree \*T)**

**{**

**if(\*T)**

**{**

**if((\*T)->lchild) /\* 有左孩子 \*/**

**DestroyBiTree(&(\*T)->lchild); /\* 销毁左孩子子树 \*/**

**if((\*T)->rchild) /\* 有右孩子 \*/**

**DestroyBiTree(&(\*T)->rchild); /\* 销毁右孩子子树 \*/**

**free(\*T); /\* 释放根结点 \*/**

**\*T=NULL; /\* 空指针赋0 \*/**

**}**

**}**

**/\* 初始条件: 二叉树T存在 \*/**

**/\* 操作结果: 若T为空二叉树,则返回TRUE,否则FALSE \*/**

**Status BiTreeEmpty(BiTree T)**

**{**

**if(T)**

**return FALSE;**

**else**

**return TRUE;**

**}**

**#define ClearBiTree DestroyBiTree**

**/\* 初始条件: 二叉树T存在。操作结果: 返回T的根 \*/**

**TElemType Root(BiTree T)**

**{**

**if(BiTreeEmpty(T))**

**return Nil;**

**else**

**return T->data;**

**}**

**/\* 初始条件: 二叉树T存在，p指向T中某个结点 \*/**

**/\* 操作结果: 返回p所指结点的值 \*/**

**TElemType Value(BiTree p)**

**{**

**return p->data;**

**}**

**/\* 给p所指结点赋值为value \*/**

**void Assign(BiTree p,TElemType value)**

**{**

**p->data=value;**

**}**

**/\* 初始条件: 二叉树T存在。操作结果: 返回T的深度 \*/**

**// Exercise 0: compute the depth of a binary tree**

**int BiTreeDepth(BiTree T)**

**{**

**int d1, d2;**

**if(!T) return 0;**

**d1=BiTreeDepth(T->lchild);**

**d2=BiTreeDepth(T->rchild);**

**return (d1>d2?d1:d2)+1;**

**}**

**/\* 按前序输入二叉树中结点的值（一个字符） \*/**

**/\* #表示空树，构造二叉链表表示二叉树T。 \*/**

**// Exercise 1: create a binary tree (symbol # means the tree is empty)**

**void CreateBiTree(BiTree \*T)**

**{**

**TElemType ch;**

**//按先序次序输入二叉树中结点的值（一个字符），'#'表示空树**

**scanf("%c",&ch); if(ch == '#') (\*T)= NULL; else {**

**\*T= (BiTree)malloc(sizeof(BiTNode)); //生成根结点**

**(\*T)->data= ch;**

**CreateBiTree(&(\*T)->lchild);**

**CreateBiTree(&(\*T)->rchild);**

**}**

**}**

**/\* 初始条件: 二叉树T存在 \*/**

**/\* 操作结果: 前序递归遍历T \*/**

**// Exercise 2: PreOrder traversal**

**void PreOrderTraverse(BiTree T)**

**{**

**if(T) {**

**printf("%c ",T->data);**

**PreOrderTraverse(T->lchild);**

**PreOrderTraverse(T->rchild);**

**}**

**}**

**/\* 初始条件: 二叉树T存在 \*/**

**/\* 操作结果: 中序递归遍历T \*/**

**// Exercise 3: InOrder traversal**

**void InOrderTraverse(BiTree T)**

**{**

**if(T) {**

**InOrderTraverse(T->lchild);**

**printf("%c ",T->data);**

**InOrderTraverse(T->rchild);**

**}**

**}**

**/\* 初始条件: 二叉树T存在 \*/**

**/\* 操作结果: 后序递归遍历T \*/**

**// Exercise 4: PostOrder traversal**

**void PostOrderTraverse(BiTree T)**

**{**

**if(T) {**

**InOrderTraverse(T->lchild);**

**InOrderTraverse(T->rchild);**

**printf("%c ",T->data);**

**}**

**}**

**int main()**

**{**

**int i;**

**BiTree T;**

**TElemType e1;**

**InitBiTree(&T);**

**StrAssign(str,"ABDH#K###E##CFI###G#J##");**

**CreateBiTree(&T);**

**printf("构造空二叉树后,树空否？%d(1:是 0:否) 树的深度=%d\n",BiTreeEmpty(T),BiTreeDepth(T));**

**e1=Root(T);**

**printf("二叉树的根为: %c\n",e1);**

**printf("\n前序遍历二叉树:");**

**PreOrderTraverse(T);**

**printf("\n中序遍历二叉树:");**

**InOrderTraverse(T);**

**printf("\n后序遍历二叉树:");**

**PostOrderTraverse(T);**

**ClearBiTree(&T);**

**printf("\n清除二叉树后,树空否？%d(1:是 0:否) 树的深度=%d\n",BiTreeEmpty(T),BiTreeDepth(T));**

**i=Root(T);**

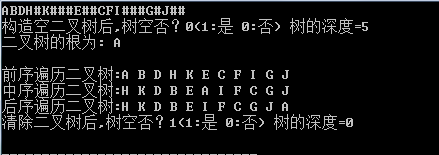
**if(!i)**

**printf("树空，无根\n");**

**return 0;**

**}**

**3.测试结果**



**4. 分析与评论**

前序后序中序遍历比较类似，计算二叉树高度，感觉自己的写法比较简洁。