《数据结构》课程实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 专业名称 | 计算机 | 年级 | 2017 | 班级 | 2 |
| 学生姓名 | 王汝芸 | 学号 | 201711010202 | 指导教师 | 郑志华 |
| 实验题目 | 实验4前序遍历创建一棵二叉树 | | | 提交时间 | 2018．4.18 |

一、实验目的和要求

例如：（1）熟悉C语言的上机环境，进一步掌握C语言的结构特点。

（2）掌握二叉树的创建方法

二、实验环境

Visual Studio2017

Windows 10

三、实验内容及实施

采用递归方法建立、遍历二叉树

**实验4：前序遍历创建一棵二叉树**

实验要求：

1. 用递归算法生成该二叉树
2. 输出前序、中序、后序遍历的遍历序列
3. 计算二叉树中结点个数

**模块图**

前序输入创建一颗二叉树

得到指针用来访问

前序中序后续遍历二叉树

……

计算二叉树中节点个数

1、建立一个二叉树；

2、前序、中序、后序输出二叉树；

3、计算二叉树节点个数

**【源程序】**

/\*实验4：前序遍历创建一棵二叉树\*/

/\*

1、 用递归算法生成该二叉树；

2、 输出前序、中序、后序遍历的遍历序列；

3、 计算二叉树中结点个数。

2018年4月18日

\*/

#include<iostream>

#include<stdlib.h>

#include<string>

#define NULLKET '#'

//二叉树节点

typedef struct tnode

{

char data;

struct tnode \*lchild, \*rchild;

}bitree, \*bitlink;

//创建二叉树函数

bitlink preCreatTree(bitlink &root)

{

char ch;

scanf\_s("%c", &ch);

if (ch == NULLKET)//NULLKEY=#

{

root = NULL;

return(root);//返回空指针，回到上一级未处理完的递归函数

}

else //子节点不为空，用递归算法创建节点

{

root = (bitree\*)malloc(sizeof(tnode)); //开辟新节点

root->data = ch; //为节点赋值

//递归创建节点

preCreatTree(root->lchild);

preCreatTree(root->rchild);

return(root);//层层返回子树根节点指针，直到最终返回树根节点

}

}

//先序遍历二叉树函数

void preorderTraversal(bitlink root)

{

if (root == NULL)

return;//如果没有儿子，返回空，回到上一级未处理完的递归函数

else

{

//输出当前节点数值

printf("%c ", root->data);

//递归先序遍历左右子树

preorderTraversal(root->lchild);

preorderTraversal(root->rchild);

}

}

//中序遍历二叉树函数

void inorderTraversal(bitlink &root)

{

if (root == NULL)

return;

else

{

inorderTraversal(root->lchild);

printf("%c ", root->data);

inorderTraversal(root->rchild);

}

}

//后序遍历二叉树函数

void postorderTraversal(bitlink &root)

{

if (root == NULL)

return;

else

{

postorderTraversal(root->lchild);

postorderTraversal(root->rchild);

printf("%c ", root->data);

}

}

//计算二叉树节点个数函数

int nodeNumber(bitlink &root)

{

int n = 0;

if (root == NULL)

{

return 0;

}

else

{

n = nodeNumber(root->lchild) + nodeNumber(root->rchild) + 1;

}

return n;

}

int main()

{

//指向树及其子树根的指针

bitlink root;

//创建二叉树

printf("请输入前序创建树：\n");

root = preCreatTree(root);//得到指向整颗树根节点的指针

//先序遍历二叉树

printf("该二叉树先序遍历结果为：\n");

preorderTraversal(root);

printf("\n");

//此时root依然指向整棵树根节点

//中序遍历二叉树

printf("该二叉树中序遍历结果为：\n");

inorderTraversal(root);

printf("\n");

//后序遍历二叉树

printf("该二叉树后序遍历结果为：\n");

postorderTraversal(root);

printf("\n");

//计算二叉树中结点个数

int n = 0;

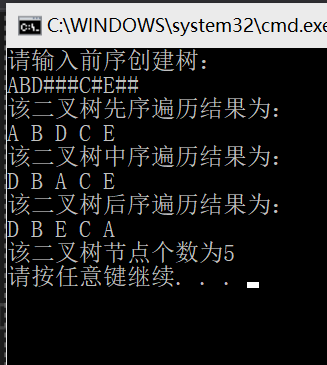
n = nodeNumber(root);

printf("该二叉树节点个数为%d\n", n);

return 0;

}

**四、实验结果 (程序的执行结果)**



**五、实验讨论（可选）**

无