实习报告：题目四

顺序表及其应用——树及其应用&二叉树的遍历

班级：17计算机一班 姓名：陈新朋 学号：1725111006 完成日期：2018.11.9

1. 需求分析

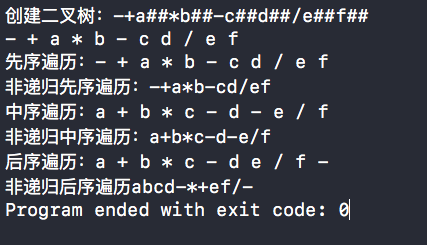
（1）以二叉链表作为存储结构，实现二叉树的先、中、后三种次序的递归与非递归遍历。

（2）测试数据以教科书图6.9的二叉树为例。

（3）设二叉树每个节点的数据均为字符，利用先序遍历序列作为输入序列创建二叉树链表存储结构，以‘#’代表节点底下无左或右分支。

（4）利用先序遍历序列先递归创建一个二叉树，在分别利用递归与非递归算法完成二叉树的遍历。

（5）测试数据及结果见图



（6）程序执行的命令为：

1）得到运算表达式；2）将运算表达式转换成后缀表达式；3）计算后缀表达式，输出结果

二、 概要设计

1. 设定二叉树的抽象数据类型定义：

ADT BiTree{

数据对象：D={a i |ai∈CharSet,i=1,2,…,n,n>=0}

数据关系：R1={<a i-1 ,a i >|a i-1 ,a i ∈D,i=2,…,n}

基本操作：

/\*\*

\* @brief 创建一个二叉树

\* 依次读入字符、‘#’代表节点的某个子节点为空

\* -T为root根节点指针

\*/

void CreateBiTree(BiTree \*T);

/\*\*

\* @brief 递归先序遍历二叉树

\* @condition 已有一个二叉树

\*/

void PreOrderTraverse(BiTree T);

/\*\*

\* @brief 递归中序遍历二叉树

\* @condition 已有一个二叉树

\*/

void InOrderTraverse(BiTree T);

/\*\*

\* @brief 递归后序遍历二叉树

\* @condition 已有一个二叉树

\*/

void PostOrderTraverse(BiTree T);

/\*\*

\* @brief 非递归先序遍历二叉树

\* @condition 已有一个二叉树

\*/

void UPreOrderTraverse(BiTree T);

/\*\*

\* @brief 非递归中序遍历二叉树

\* @condition 已有一个二叉树

\*/

void UInOrderTraverse(BiTree T);

/\*\*

\* @brief 非递归后序遍历二叉树

\* @condition 已有一个二叉树

\*/

void UPostOrderTraverse(BiTree T);

}}ADT list

1. 详细设计

//binary tree.cpp

#include "binary tree.hpp"

void CreateBiTree(BiTree \*T)

{

char ch;

cin>>ch;

if(ch=='#')

T=NULL;

else

{

\*T=new BitNode;

(\*T)->data=ch;

(\*T)->lc=NULL;

(\*T)->rc=NULL;

cout<<(\*T)->data<<' ';

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

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

}

}

void PreOrderTraverse(BiTree T)

{

if(T)

{

cout<<T->data<<' ';

PreOrderTraverse(T->lc);

PreOrderTraverse(T->rc);

}

}

void InOrderTraverse(BiTree T)

{

if(T!=NULL)

{

InOrderTraverse(T->lc);

cout<<T->data<<' ';

InOrderTraverse(T->rc);

}

}

void PostOrderTraverse(BiTree T)

{

if(T)

{

InOrderTraverse(T->lc);

InOrderTraverse(T->rc);

cout<<T->data<<' ';

}

}

void UPreOrderTraverse(BiTree T)

{

BiTree stack[50];

int top;

BitNode \*p;

top=0;

p=T;

while(p!=NULL||top>0)

{

while(p!=NULL)

{

cout<<p->data;

stack[top++]=p;

p=p->lc;

}

if(top>0)

{

p=stack[--top];

p=p->rc;

}

}

}

void UInOrderTraverse(BiTree T)

{

BiTree stack[50];

int top;

BitNode \*p;

top=0;

p=T;

while(p!=NULL||top>0)

{

while(p!=NULL)

{

stack[top++]=p;

p=p->lc;

}

if(top>0)

{

p=stack[--top];

cout<<p->data;

p=p->rc;

}

}

}

void UPostOrderTraverse(BiTree T)

{

BiTree stack[50];

int top;

BitNode \*p,\*q;

top=0;

p=T;

while(p!=NULL||top>0)

{

while(p!=NULL)

{

stack[top++]=p;

p=p->lc;

}

if(top>0)

{

p=stack[top-1];

if(p->rc==NULL||p->rc==q)

{

cout<<p->data;

q=p;

p=NULL;

top--;

}

else

{

p=p->rc;

}

}

}

}

//main.cpp

#include <iostream>

#include "binary tree.hpp"

using namespace std;

int main() {

BitNode tree;

BiTree P=&tree;

cout<<"创建二叉树：";

CreateBiTree(&P);

cout<<endl<<"先序遍历：";

PreOrderTraverse(P);

cout<<endl<<"非递归先序遍历：";

UPreOrderTraverse(P);

cout<<endl<<"中序遍历：";

InOrderTraverse(P);

cout<<endl<<"非递归中序遍历：";

UInOrderTraverse(P);

cout<<endl<<"后序遍历：";

PostOrderTraverse(P);

cout<<endl<<"非递归后序遍历";

UPostOrderTraverse(P);

cout<<endl;

return 0;

}

四、调试分析

1．创建二叉树的时候遇到了野指针问题，发现是自己没有给结构体变量动态分配内存。

2．先序遍历的时候发现没有显示出来，原因是函数用的指针，运行完就销毁了，所以先序遍历时输出不了值。改成应用或者return结构体指针的函数即可

3．先序遍历时输出的数值是正确的，但是中序遍历和后序遍历都错了，发现是自己ctrl+V后忘记修改递归函数内部的函数名了。

4．算法复杂度分析。递归遍历：时间复杂度为O（n），空间复杂度为O（n）。

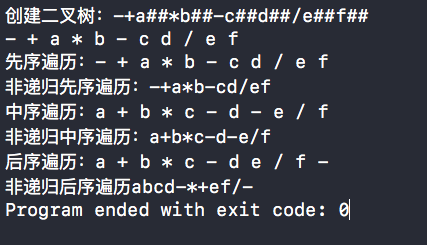
5．经验体会：程序尽量自己手打，少用复制黏贴，避免低级错误。

五、用户使用说明

运行程序后，显示“创建二叉树：”，此时，用户输入先序遍历顺序的序列，以‘#’代表空。

回车，显示创建结果以及所有遍历的结果。

六、测试结果



七、 附录

源程序文件名清单：

main.cpp//主程序

binary tree.cpp//源程序

binary tree.hpp//头文件