\*\*\*\*\*\*\*\*\*\*二叉树（基于二叉链表）实现\*\*\*\*\*\*\*\*\*\*

1. **节点类(Binode.h)：**

typedef char E;

class BiNode{

private:

E it; // The node's value

BiNode\* lc; // Pointer to left child

BiNode\* rc; // Pointer to right child

public:

BiNode() { lc = rc = NULL; }

BiNode(E e, BiNode\* l,BiNode\* r) { it = e; lc = l; rc = r; }

E& element() { return it; }

void setElement(E& e) { it = e; }

//返回左孩子

BiNode\* left() { return lc; }

void setLeft(BiNode\* b) { lc = (BiNode\*)b; }

//返回右孩子

BiNode\* right() { return rc; }

void setRight(BiNode\* b) { rc = (BiNode\*)b; }

//判断是否为叶子节点

bool isLeaf() { return (lc == NULL) && (rc == NULL); }

};

1. **树类（BinTree.h）：**

#include <iostream>

using namespace std;

class BinTree:public BiNode{

private:

char c;

BiNode\* root; // Root of the BinTree BinTree树的根结点

public:

BiNode\* Root(){

return root;

}

void setroot(BiNode\* n){

root=n;

}

//判断二叉树是否为空

bool BiTreeEmpty(BiNode\* rt)

{

if(rt!=NULL) return true;

else return false;

}

//前序遍历

void preorder(BiNode\* rt)

{

if(rt!=NULL)

{

cout<<rt->element()<<" ";

preorder(rt->left());

preorder(rt->right());

}

}

//中序遍历

void Inorder(BiNode\* rt)

{

if(rt!=NULL)

{

Inorder(rt->left());

cout<<rt->element()<<" ";

Inorder(rt->right());

}

}

//后序遍历

void postorder(BiNode\* rt)

{

if(rt!=NULL)

{

postorder(rt->left());

postorder(rt->right());

cout<<rt->element()<<" ";

}

}

//层次遍历

void LevelOrderTraverse(BiNode\* rt)

{

queue<BiNode\*> q;

if(rt!=NULL)

q.push(rt);

BiNode\* b;

while(!q.empty())

{

b=q.front();

cout<<b->element()<<' ';

q.pop();

if(b->left())

q.push(b->left());

if(b->right())

q.push(b->right());

}

}

//二叉树深度

int BiTreeDepth(BiNode\* rt)

{

int lh=0,rh=0 ;

if(rt!=NULL)

{

lh=BiTreeDepth(rt->left());

rh=BiTreeDepth(rt->right());

return (lh>rh?lh:rh)+1;

}

else return 0 ;

}

//节点数统计

int count(BiNode\* rt)

{

if(rt) return count(rt->left())+count(rt->right())+1;

else return 0;

}

};

1. **测试类（BinTree.cpp）：**

#include <iostream>

#include <assert.h>

#include <stdlib.h>

#include <string.h>

#include <queue>

#include "BiNode.h"

#include "BinTree.h"

using namespace std;

typedef char E;

char c[50];

int count=0;

string s;

BiNode\* r;

BinTree T;

//建树

BiNode\* CreateBiTree(char ch){

BiNode\* rt;

if(ch=='#') rt=NULL;

else{

rt=new BiNode;

rt->setElement(ch);

ch=c[++count];

rt->setLeft(CreateBiTree(ch));

ch=c[++count];

rt->setRight(CreateBiTree(ch));

}

//设置根节点

T.setroot(rt);

return rt;

}

int main()

{

char ch;

int depth;

int num;

printf("please input the tree: ");

cout<<"以前序遍历的顺序输入，空节点以#代替,例AB#D##CE##F##"<<endl;

cin>>s;

strcpy(c,s.c\_str());

r=CreateBiTree(c[0]);

cout<<"判断树是否是空树：";

if(T.BiTreeEmpty(r))

cout<<"No"<<endl;

else cout<<"Yes"<<endl;

cout<<"前序遍历：";

T.preorder(r);

cout<<endl;

cout<<"中序遍历：";

T.Inorder(r);

cout<<endl;

cout<<"后序遍历：";

T.postorder(r);

cout<<endl;

cout<<"层次遍历：";

T.LevelOrderTraverse(r);

cout<<endl;

cout<<"树的深度：";

depth=T.BiTreeDepth(r);

cout<<depth<<endl;

cout<<"树的结点个数：";

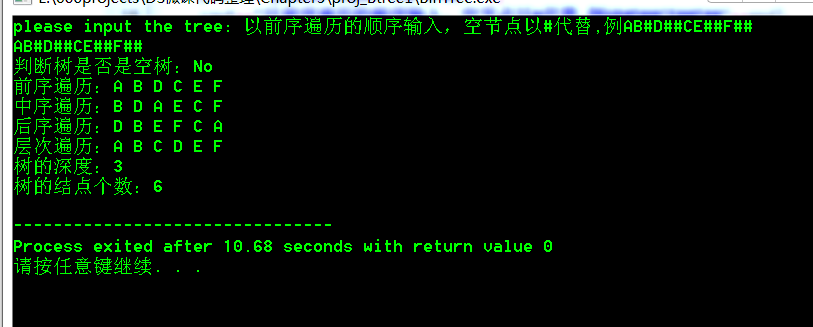
num=T.count(r);

cout<<num<<endl;

return 0;

}

**测试结果示例：**

****

**【附录】**

版本信息声明：

Dev-C++ 5.11

TDM-GCC 4.9.2 64-bit Release

资料整理人：  
物联网工程1402班 吴彦彦

物联网工程1402班 宁静仪