```
#include<iostream>
using namespace std;
class binarytreenode//二叉树结点
{
	int data;
public:
	binarytreenode * leftchild;
	binarytreenode * rightchild;
	binarytreenode * next;
	binarytreenode(){};
	binarytreenode(int & d)
	{
		data=d;
		leftchild=NULL;
		rightchild=NULL;
	}
	int& get_data(){	return data;	}
	~binarytreenode(){};
};
//栈
class stack
{
private:
	int size;
	int top;
	binarytreenode** ar;
public:
	stack(int size)
	{
		this->size=size;
		top=-1;
```

```
        ar=new binarytreenode*[size];
}
bool push(binarytreenode* item)//入栈
{
        if(top==size-1)
        {
                cout<<"栈已满!"<<endl;
                return false;
        }
        else
        {
                ar[++top]=item;
                return true;
        }
}
bool pop()//出栈
{
        if(top==-1)
        {
                cout<<"栈为空!"<<endl;
                return false;
        }
        else
        {
                top--;
                return true;
        }
}
binarytreenode* read()//读取栈顶元素
{
        if(top==-1)
        {
            cout<<"栈为空!"<<endl;
            return NULL;
        }
        return ar[top];
    }
    bool empty()
    {
        if(top==-1)
            return true;
        return false;
    }
    ~stack(){};
};
//二叉树
class binarytree
{
    binarytreenode * root;
    int size;
public:
    binarytree()
    {   size=0;
        root=NULL;      }
    int get_size(){ return size;    }
    void change_size(int&n){        size=n; }
    void change_root(binarytreenode*p){     root=p; }
    binarytreenode* get_root(){     return root;    }
    void postorder()//后序遍历
    {
        binarytreenode * p = get_root(),*prev=NULL;
        stack st(get_size());
        cout<<"后序遍历结果: ";
        if(p==NULL)
            cout<<"二叉树为空!";
        while(p!=NULL)
        {
            for(;p->leftchild!=NULL;p=p->leftchild)
                st.push(p);
            while(p!=NULL && (p->rightchild==NULL||p->rightchild==prev))//右子树不存在或已经访问过,访问该结点
            {
                cout<<p->get_data()<<" ";
                prev=p;
                if(st.empty())
                    goto last;
                p=st.read();//读取栈顶元素
                st.pop();
            }
            st.push(p);
            p=p->rightchild;//访问右子树
        }
        last:
            cout<<endl;
    }
    binarytreenode* pre_in_postorder(int pre[],int in[],int length)//由先序和中序序列确定二叉树
    {
        if(length==0)
            return NULL;
        binarytreenode*p;
        p=new binarytreenode(pre[0]);
        int i=0;//标记当前根节点下标
        for(i=0;i<length;i++)//标记当前根节点下标
            if(in[i]==p->get_data())
                break;
        if(i!=0)
        {
            int*ppre,*iin;
            ppre=new int[i];
            iin=new int[i];
            for(int j=0;j<i;j++)
            {
                ppre[j]=pre[j+1];
                iin[j]=in[j];
            }
            p->leftchild=pre_in_postorder(ppre,iin,i);
        }
        if(i!=length-1)
        {
            int*ppre,*iin;
            ppre=new int[length-i-1];
            iin=new int[length-i-1];
            for(int j=0;j<length-i-1;j++)
            {
                ppre[j]=pre[j+1+i];
                iin[j]=in[j+1+i];
            }
            p->rightchild=pre_in_postorder(ppre,iin,length-i-1);
        }
        return p;
    }//bool pre_in_postorder()
    int get_height(binarytreenode*p)//统计高度
    {
        if(p->leftchild==NULL && p->rightchild==NULL)//叶子
            return 1;
        else if(p->leftchild!=NULL && p->rightchild==NULL)
            return 1+get_height(p->leftchild);
        else if(p->rightchild!=NULL && p->leftchild==NULL)
            return 1+get_height(p->rightchild);
        else if(p->leftchild!=NULL && p->rightchild!=NULL)
        {
            int i1=1+get_height(p->leftchild);
            int i2=1+get_height(p->rightchild);
            return (i1>i2)?i1:i2;
        }
    }
    void get_width(binarytreenode*p,int i,int wide[])//统计各层结点数
    {
        wide[i++]++;
        if(p->leftchild!=NULL)
            get_width(p->leftchild,i,wide);
        if(p->rightchild!=NULL)
            get_width(p->rightchild,i,wide);
    }
};
int main()
{
    binarytree tree;
    int length=0;
    cout<<"输入序列长度:";
    cin>>length;
    int * preorder,* inorder;
    preorder=new int[length];
    inorder=new int[length];
```

```
cout<<"输入先序序列: ";
for(int i=0;i<length;i++)
        cin>>preorder[i];
cout<<"输入中序序列: ";
for(int i=0;i<length;i++)
        cin>>inorder[i];
cout<<"创建中,Wait. . ."<<endl;
tree.change_root(tree.pre_in_postorder(preorder,inorder,length));
tree.change_size(length);
tree.postorder();//后序输出
int height=tree.get_height(tree.get_root()),*wide;//统计高度
wide=new int[height];//宽度
for(int j=0;j<height;j++)
        wide[j]=0;//宽度初始化
tree.get_width(tree.get_root(),0,wide);//统计宽度
//判断宽度
int tag=1,i=0;
for(;i<height;i++)
{
        if(wide[i]!=tag)
                break;
        tag*=2;
}

if(i<height-1)
{
        cout<<"不是完全二叉树,最后一层结点之前的层结点不满!"<<endl;
        return 0;
}
int last=tree.get_size()-tag+1;
binarytreenode*p=tree.get_root();
```

```
	int flag=0,j=0;
	for(int k=1;k<=last;k++)//1<=k<=3
	{
		j=0;
		p=tree.get_root();
		for(;j<height-k;j++)
			p=p->leftchild;
		for(;j<k-1;j++)
			p=p->rightchild;
		if(p==NULL)
		{
			flag=1;
			break;
		}
	}
	if(flag==1)
	{
		cout<<"不是完全二叉树，最后一层结点不是自左向右排列!"<<endl;
		return 0;
	}
	cout<<"是完全二叉树!"<<endl;
	return 0;
}
```