/\*

\*数据结构（C++）

\*二叉树的相关操作

\*二叉树的建立

\*二叉树的有序建立

\*二叉树的先序、中序、后序遍历递归实现以及非递归实现算法

\*二叉树的层序遍历

\*求二叉树的高度、节点个数、叶子节点数

\*二叉树的销毁

\*样例输入ab##cdd#e###,先序的方式，遇到空指针就是输入#

\*/

#include <iostream>

#include<stack>

#include<queue>

using namespace std;

typedef char ElementType;

typedef struct Node

{

int data;

Node\*next;

}Node;

typedef struct Leaf

{

ElementType data;

Leaf\*left;

Leaf\*right;

}Leaf;

//二叉树的建立

Leaf\*create()

{

Leaf \*p=NULL;

ElementType temp;

cin>>temp;

if('#'==temp)

return NULL;

p=new Leaf;

p->data=temp;

p->left=create();

p->right=create();

return p;

}

//二叉树的插入排序,按字母由小到大的顺序(中序输出为从小到大的顺序)（样例输入fabcwrzoutd#）

Leaf\*createByorder()

{

Leaf\*head=NULL;

ElementType data;

cout<<"请输入:（按#结束输入）\n";

while(cin>>data)

{

if(data=='#')

break;

Leaf\*p=new Leaf;

p->data=data;

p->left=NULL;

p->right=NULL;

if(NULL==head)

{

head=p;

}

else

{

Leaf\*q=head;

while(q!=NULL)

{

if(p->data<q->data)

{

if(!q->left)

{

q->left=p;

break;

}

q=q->left;

}

else

{

if(!q->right)

{

q->right=p;

break;

}

q=q->right;

}

}

}

}

return head;

}

//二叉树的先序遍历

void firstOrder(Leaf\*head)

{

if(!head)

return;

cout<<head->data<<" ";

firstOrder(head->left);

firstOrder(head->right);

}

//二叉树先序遍历非递归实现

void firstOrderNot(Leaf\*head)

{

stack<Leaf\*>s;

s.push(head);

while(!s.empty())

{

Leaf\*temp=s.top();

s.pop();

if(temp)

{

cout<<temp->data<<" ";

s.push(temp->right);

s.push(temp->left);

}

}

}

//二叉树的中序遍历

void middleOrder(Leaf \*head)

{

if(!head)

return;

middleOrder(head->left);

cout<<head->data<<" ";

middleOrder(head->right);

}

void inOrder2(Leaf \*root) //非递归中序遍历（网上其他人的实现方法）

{

stack<Leaf\*> s;

Leaf \*p=root;

while(p!=NULL||!s.empty())

{

while(p!=NULL)

{

s.push(p);

p=p->left;

}

if(!s.empty())

{

p=s.top();

cout<<p->data<<" ";

s.pop();

p=p->right;

}

}

}

//二叉树的中序遍历非递归实现

void middleOrderNot(Leaf\*head)

{

if(!head)

return;

stack<Leaf\*>s;

s.push(head);

Leaf\*q=NULL;

bool flag=true;

while (!s.empty())

{

q=s.top();

while (q->left&&flag)

{

s.push(q->left);

q=q->left;

}

flag=false;

Leaf\*temp=s.top();

s.pop();

if(temp){

cout<<temp->data<<" ";

if(temp->right)

{

s.push(temp->right);

flag=true;

}

}

}

}

//二叉树的后序遍历

void lastOrder(Leaf\*head)

{

if(!head)

return;

lastOrder(head->left);

lastOrder(head->right);

cout<<head->data<<" ";

}

//二叉树的后序遍历非递归实现

void lastOrderNot(Leaf\*head)

{

if(!head)

return;

stack<Leaf\*>s;

Leaf\*p=head;

Leaf\*visted=NULL;

while(p!=NULL||(!s.empty()))

{

while(p!=NULL)

{

s.push(p);

p=p->left;

}

p=s.top();

if(p->right==NULL||p->right==visted)

{

cout<<p->data<<" ";

s.pop();

visted=p;

p=NULL;

}

else

{

p=p->right;

}

}

}

//二叉树层序遍历

void floorOrder(Leaf\*head)

{

if(!head)

return;

queue<Leaf\*> q;

q.push(head);

while (!q.empty())

{

Leaf\*temp=q.front();

q.pop();

if(temp)

{

cout<<temp->data<<" ";

q.push(temp->left);

q.push(temp->right);

}

}

}

//求二叉树的高度

int gettHeight(Leaf \*head)

{

if(!head)

return 0;

int left=1+gettHeight(head->left);

int right=1+gettHeight(head->right);

return left>right?left:right;

}

//二叉树的节点个数

int getCount(Leaf\*head)

{

if(!head)

return 0;

int left=getCount(head->left);

int right=getCount(head->right);

return left+right+1;

}

//二叉树的叶子节点个数

int getLeafCount(Leaf\*head)

{

static int count=0;

if(!head)

return 0;

if((!head->left)&&(!head->right))

count++;

getLeafCount(head->left);

getLeafCount(head->right);

return count;

}

//二叉树内存空间释放

void destroy(Leaf\*&head)

{

if(!head)

return;

destroy(head->left);

destroy(head->right);

delete(head);

head=NULL;

}

//int main()

//{

// cout<<"按字母从小到大的顺序建立二叉树：\n";

// Leaf\*head=createByorder();

// //cout<<"请输入二叉树，如果是空指针，请用#代替"<<endl;

// //Leaf \*head=create();

// cout<<"\n二叉树的先序遍历输出:";

// firstOrder(head);

// cout<<"\n二叉树的先序遍历非递归输出:";

// firstOrderNot(head);

// cout<<"\n二叉树的中序遍历输出:";

// middleOrder(head);

// cout<<"\n二叉树的中序遍历非递归输出:";

// middleOrderNot(head);

// cout<<"\n二叉树的后序遍历输出:";

// lastOrder(head);

// cout<<"\n二叉树后序遍历非递归输出:";

// lastOrderNot(head);

// cout<<"\n二叉树的层序遍历:";

// floorOrder(head);

// cout<<"\n二叉树的高度为:";

// cout<<gettHeight(head);

// cout<<"\n二叉树的节点个数为:";

// cout<<getCount(head);

// cout<<"\n二叉树的叶子节点个数为:";

// cout<<getLeafCount(head);

// cout<<"\n二叉树正在销毁。。。。";

// destroy(head);

// if(!head)

// cout<<"\n二叉树已经销毁\n";

// return 0;

//}