#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

#include<assert.h>

//二叉树的结点

typedef char TreeDateType;

typedef struct Tree

{

TreeDateType date;

struct Tree\* left;

struct Tree\* right;

}Tree;

//队列

typedef struct QueueNode

{

Tree\* Qdata;

struct QueueNode\* prev;

struct QueueNode\* next;

}QueueNode;

void InitQueue(QueueNode\*\* pphead)

{

\*pphead = (QueueNode\*)malloc(sizeof(QueueNode));

(\*pphead)->Qdata = NULL;

(\*pphead)->next = \*pphead;

(\*pphead)->prev = \*pphead;

}

void PushQeueNode(QueueNode\*\* pphead, Tree\* X)

{

QueueNode\* tail = (\*pphead)->prev;

QueueNode\* newnode = (QueueNode\*)malloc(sizeof(QueueNode));

newnode->Qdata = X;

newnode->prev = tail;

tail->next = newnode;

newnode->next = (\*pphead);

(\*pphead)->prev = newnode;

}

void PrinQueue(QueueNode\* phead)

{

QueueNode\* cur = phead->next;

while (phead != cur)

{

printf("%c ", cur->Qdata->date);

cur = cur->next;

}

}

Tree\* GetQueueNode(QueueNode\* phead)

{

QueueNode\* frist = phead->next;

QueueNode\* second = frist->next;

Tree\* num = frist->Qdata;

phead->next = second;

second->prev = phead;

return num;

}

bool JaEmpty(QueueNode\* phead)

{

return phead->next == phead;

}

Tree\* CreatTree()

{

char a = 0;

scanf("%c", &a);

if (a == ' ')

{

return NULL;

}

Tree\* root = (Tree\*)malloc(sizeof(Tree));

root->date = a;

root->left = CreatTree(root->left);

root->right = CreatTree(root->right);

return root;

}

//先序遍历

void PrintPrev(Tree\* root)

{

if (root == NULL)

{

return;

}

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

PrintPrev(root->left);

PrintPrev(root->right);

}

//后序遍历

void PrintBank(Tree\* root)

{

if (root==NULL)

{

return;

}

PrintBank(root->left);

PrintBank(root->right);

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

}

//中序遍历

void PrintMid(Tree\* root)

{

if (root==NULL)

{

return;

}

PrintMid(root->left);

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

PrintMid(root->right);

}

//层序遍历

void LayerSequence(Tree\* T, QueueNode\* phead)

{

Tree\* cur = T;

PushQeueNode(&phead, cur);

while (!JaEmpty(phead))

{

Tree\* Node = GetQueueNode(phead);

printf("%c ", Node->date);

if (Node->left != NULL)

{

PushQeueNode(&phead, Node->left);

}

if (Node->right != NULL)

{

PushQeueNode(&phead, Node->right);

}

}

}

//二叉树的最大深度

int maxDepth(Tree\* root) {

if (root == NULL)

{

return 0;

}

//return maxDepth(root->left)>maxDepth(root->right)?maxDepth(root->left)+1:maxDepth(root->right)+1;

int depthleft = maxDepth(root->left);

int depthright = maxDepth(root->right);

return depthleft > depthright ? depthleft + 1 : depthright + 1;

}

//求树的结点个数

int NodeSize(Tree\* root)

{

if (root == NULL)

{

return 0;

}

return NodeSize(root->left) + NodeSize(root->right) + 1;

}

//求叶子节点数

int LeafNodeSize(Tree\* root)

{

if (root == NULL)

{

return 0;

}

if (root->left == NULL && root->right == NULL)

{

return 1;

}

return LeafNodeSize(root->left) + LeafNodeSize(root->right);

}

int main()

{

QueueNode\* Queue =NULL;

InitQueue(&Queue);

Tree\* root = CreatTree();

printf("前序:");

PrintPrev(root);

printf("\n");

printf("后序:");

PrintBank(root);

printf("\n");

printf("中序:");

PrintMid(root);

printf("\n");

printf("层序:");

LayerSequence(root, Queue);

printf("\n");

printf("叶子节点数:%d\n", LeafNodeSize(root));

printf("最大深度%d\n", maxDepth(root));

printf("节点数%d\n", NodeSize(root));

return 0;

}

