#include<stdio.h>

#include<stdlib.h>

typedef struct Tree {

int data; // 存放数据域

struct Tree\* lchild; // 遍历左子树指针

struct Tree\* rchild; // 遍历右子树指针

}Tree, \* BitTree;

BitTree CreateLink()

{

int data;

int temp;

BitTree T;

scanf\_s("%d", &data); // 输入数据

temp = getchar(); // 吸收空格

if (data == -1) { // 输入-1 代表此节点下子树不存数据，也就是不继续递归创建

return NULL;

}

else {

T = (BitTree)malloc(sizeof(Tree)); // 分配内存空间

if (T == NULL)

printf("内存分配失败！");

else T->data = data; // 把当前输入的数据存入当前节点指针的数据域中

printf("请输入%d的左子树: ", data);

T->lchild = CreateLink(); // 开始递归创建左子树

printf("请输入%d的右子树: ", data);

T->rchild = CreateLink(); // 开始到上一级节点的右边递归创建左右子树

return T; // 返回根节点

}

}

// 先序遍历

void ShowXianXu(BitTree T) // 先序遍历二叉树

{

if (T == NULL) // 递归中遇到NULL，返回上一层节点

{

return;

}

printf("%d ", T->data);

ShowXianXu(T->lchild); // 递归遍历左子树

ShowXianXu(T->rchild); // 递归遍历右子树

}

// 中序遍历

void ShowZhongXu(BitTree T) // 先序遍历二叉树

{

if (T == NULL) // 递归中遇到NULL，返回上一层节点

{

return;

}

ShowZhongXu(T->lchild); // 递归遍历左子树

printf("%d ", T->data);

ShowZhongXu(T->rchild); // 递归遍历右子树

}

// 后序遍历

void ShowHouXu(BitTree T) // 后序遍历二叉树

{

if (T == NULL) // 递归中遇到NULL，返回上一层节点

{

return;

}

ShowHouXu(T->lchild); // 递归遍历左子树

ShowHouXu(T->rchild); // 递归遍历右子树

printf("%d ", T->data);

}

int main()

{

BitTree S;

printf("请输入第一个节点的数据:\n");

S = CreateLink(); // 接受创建二叉树完成的根节点

printf("先序遍历结果: \n");

ShowXianXu(S); // 先序遍历二叉树

printf("\n中序遍历结果: \n");

ShowZhongXu(S); // 中序遍历二叉树

printf("\n后序遍历结果: \n");

ShowHouXu(S); // 后序遍历二叉树

return 0;

}