递归

```
#include <stdio.h>
#include <string.h>
#define TElemType int
//构造结点的结构体
typedef struct BiTNode{
    TElemType data;//数据域
    struct BiTNode *lchild,*rchild;//左右孩子指针
}BiTNode,*BiTree;
//初始化树的函数
void CreateBiTree(BiTree *T){
    *T=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->data=1;
    (*T)->lchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->rchild=(BiTNode*)malloc(sizeof(BiTNode));

    (*T)->lchild->data=2;
    (*T)->lchild->lchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->lchild->rchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->lchild->rchild->data=5;
    (*T)->lchild->rchild->lchild=NULL;
    (*T)->lchild->rchild->rchild=NULL;
    (*T)->rchild->data=3;
    (*T)->rchild->lchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->rchild->lchild->data=6;
    (*T)->rchild->lchild->lchild=NULL;
    (*T)->rchild->lchild->rchild=NULL;
    (*T)->rchild->rchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->rchild->rchild->data=7;
    (*T)->rchild->rchild->lchild=NULL;
    (*T)->rchild->rchild->rchild=NULL;
    (*T)->lchild->lchild->data=4;
    (*T)->lchild->lchild->lchild=NULL;
    (*T)->lchild->lchild->rchild=NULL;
}

//模拟操作结点元素的函数，输出结点本身的数值
void displayElem(BiTNode* elem){
    printf("%d ",elem->data);
}
//先序遍历
void PreOrderTraverse(BiTree T){
    if (T) {
        displayElem(T);//调用操作结点数据的函数方法
        PreOrderTraverse(T->lchild);//访问该结点的左孩子
```

```
        PreOrderTraverse(T->rchild);//访问该结点的右孩子
    }
    //如果结点为空，返回上一层
    return;
}
int main() {
    BiTree Tree;
    CreateBiTree(&Tree);
    printf("先序遍历: \n");
}
```

```
#include <stdio.h>
#include <string.h>
#define TElemType int
int top=-1;//top变量时刻表示栈顶元素所在位置
//构造结点的结构体
typedef struct BiTNode{
    TElemType data;//数据域
    struct BiTNode *lchild,*rchild;//左右孩子指针
}BiTNode,*BiTree;
//初始化树的函数
void CreateBiTree(BiTree *T){
    *T=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->data=1;
    (*T)->lchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->rchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->lchild->data=2;
    (*T)->lchild->lchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->lchild->rchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->lchild->rchild->data=5;
    (*T)->lchild->rchild->lchild=NULL;
    (*T)->lchild->rchild->rchild=NULL;
    (*T)->rchild->data=3;
    (*T)->rchild->lchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->rchild->lchild->data=6;
    (*T)->rchild->lchild->lchild=NULL;
    (*T)->rchild->lchild->rchild=NULL;
    (*T)->rchild->rchild=(BiTNode*)malloc(sizeof(BiTNode));
    (*T)->rchild->rchild->data=7;
    (*T)->rchild->rchild->lchild=NULL;
    (*T)->rchild->rchild->rchild=NULL;
    (*T)->lchild->lchild->data=4;
    (*T)->lchild->lchild->lchild=NULL;
```

```
    (*T)->lchild->lchild->rchild=NULL;
}
//前序遍历使用的进栈函数
void push(BiTNode** a,BiTNode* elem){
    a[++top]=elem;
}
//弹栈函数
void pop( ){
    if (top==-1) {
        return ;
    }
    top--;
}
//模拟操作结点元素的函数，输出结点本身的数值
void displayElem(BiTNode* elem){
    printf("%d ",elem->data);
}
//拿到栈顶元素
BiTNode* getTop(BiTNode**a){
    return a[top];
}
//先序遍历非递归算法
void PreOrderTraverse(BiTree Tree){
    BiTNode* a[20];//定义一个顺序栈
    BiTNode * p;//临时指针
    push(a, Tree);//根结点进栈
    while (top!=-1) {
        p=getTop(a);//取栈顶元素
        pop();//弹栈
        while (p) {
            displayElem(p);//调用结点的操作函数
            //如果该结点有右孩子，右孩子进栈
            if (p->rchild) {
                push(a,p->rchild);
            }
            p=p->lchild;//一直指向根结点最后一个左孩子
        }
    }
}
int main(){
    BiTree Tree;
    CreateBiTree(&Tree);
    printf("先序遍历: \n");
    PreOrderTraverse(Tree);
}
```

1 2 4 5 3 6 7