```
#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);
}
//中序遍历
void INOrderTraverse(BiTree T){
    if (T) {
        INOrderTraverse(T->lchild);//遍历左孩子
        displayElem(T);//调用操作结点数据的函数方法
        INOrderTraverse(T->rchild);//遍历右孩子
```

```
    }
    //如果结点为空，返回上一层
    return;
}

int main() {
    BiTree Tree;
    CreateBiTree(&Tree);
    printf("中序遍历算法: \n");
    INOrderTraverse(Tree);
}
```

```
//非递归
int INOrderTraverse(BiTree T){
      BiTree  stack[20];
      BiTNode *temp;
      int top,base;
      base=top=0;
      stack[++top]=T;
      while(top!=base){
            temp=stack[top];
            while(temp){
                  temp=temp->lchild;
                  stack[++top]=temp;

            }
            top--;
            if(top!=base){
                  temp=stack[top--];
                  printf("%d",temp->data);

                  stack[++top]=temp->rchild;

            }
      }
}
```

4 2 5 1 6 3 7