数据结构实验报告
张恒硕　　2212266　　智科 2 班

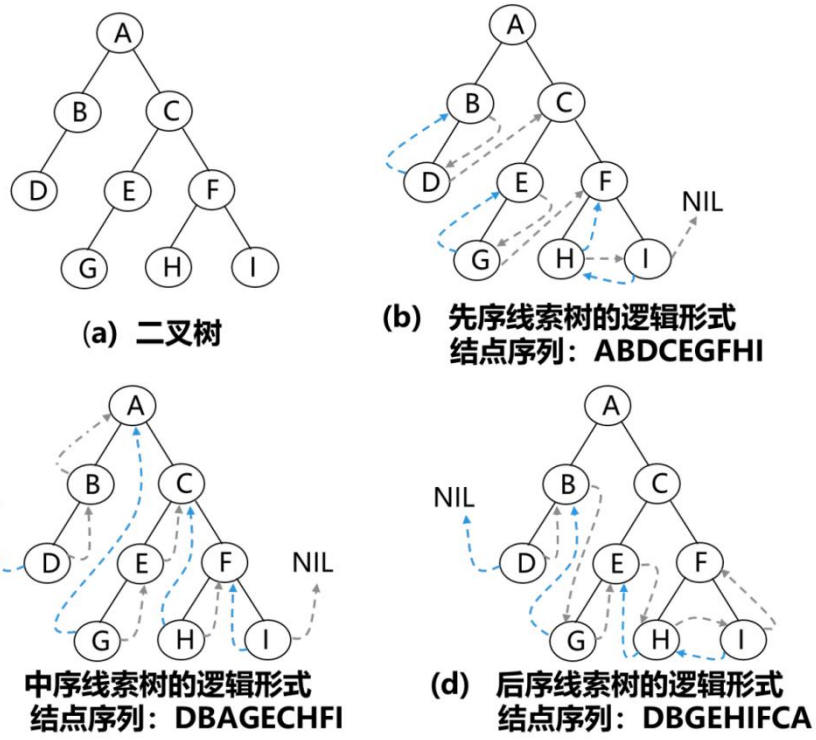
实验目标：　实现线索化二叉树（输入二叉链表存储的二叉树，输出线索化后的二叉树）。实现针对此线索化二叉树的先序和后序遍历算法。

声明：以下代码中，链式树是上节课自己写的，线索树是网课（同学推荐的）学的。

线索化二叉树

实验原理：

线索化二叉树是一种对二叉树进行的预处理，可以在 O(1)的时间复杂度内实现二叉树的遍历。在线索化二叉树中，每个节点都有前驱和后继节点（头尾各没有其中一个）。根据输出方式的不同和子树的有无，前驱后继指向不同节点。



代码：

```
#include<iostream>
using namespace std;
class tree { //链式树类
public:
    int value; //元素
    tree* left; //左枝
    tree* right; //右枝
    tree() { //无参构造
        value = 0;
```

```
        left = right = NULL;
    }
    tree(int x) { //构造叶子结点
        value = x;
        left = right = NULL;
    }
    void set(int a, int x) { //引申结点，左为1，右为2
        if (a == 1) {
            left = new tree(x);
        }
        if (a == 2) {
            right = new tree(x);
        }
    }
};
void Preorder(tree* root) { //前序输出
    if (root == NULL) {
        return;
    }
    cout << root->value << " ";
    Preorder(root->left);
    Preorder(root->right);
}
void Inorder(tree* root) { //中序输出
    if (root == NULL) {
        return;
    }
    Inorder(root->left);
    cout << root->value << " ";
    Inorder(root->right);
}
void Postorder(tree* root) { //后序输出
    if (root == NULL) {
        return;
    }
    Postorder(root->left);
    Postorder(root->right);
    cout << root->value << " ";
}
class node { //线索树结点类
public:
    char date; //数据
    int ltag, rtag; //左右标识
    node* lchild; //左孩子
```

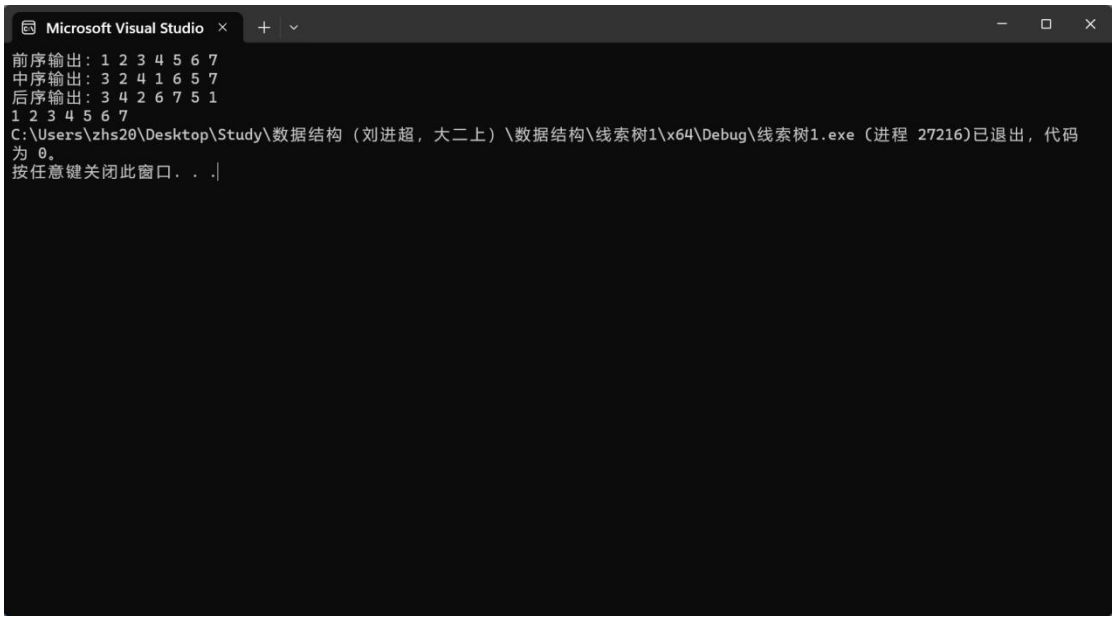
```
    node* rchild; //右孩子
    node() { //无参构造
        date = 0;
        ltag = rtag = 0;
        lchild = rchild = NULL;
    }
    node(int number) { //有参构造
        date = number;
        ltag = rtag = 0;
        lchild = rchild = NULL;
    }
};
node* pre = NULL; //前驱
void creatprethread(node*& t) { //先序线索化
    pre = NULL;
    if (t != NULL) {
        prethread(t);
        if (pre->rchild == NULL) { //处理尾结点
            pre->rtag = 1;
        }
    }
}
void prethread(node*& t) { //先序遍历设定左右标识
    if (t != NULL) {
        if (t->lchild == NULL) {
            t->lchild = pre;
            t->ltag = 1;
        }
        if (pre != NULL && pre->rchild == NULL) {
            pre->rchild = t;
            pre->rtag = 1;
        }
        pre = t;
        if (t->ltag == 0) {
            prethread(t->lchild);
        }
        prethread(t->rchild);
    }
}
void prevoder(node* t) { //遍历。从根节点开始，存在左孩子，就遍历左孩子，否则遍历右孩子，
如此循环直到遍历完所有节点。
    node* p = t;
    while (p != NULL) {
        cout << p->date;
```

```
            if (p->ltag == 1) {
                p = p->lchild;
            }
            else p = p->rchild;
        }
}
int main() {
    tree* root = new tree(1);
    root->left = new tree(2);
    root->left->left = new tree(3);
    root->left->right = new tree(4);
    root->right = new tree(5);
    root->right->left = new tree(6);
    root->right->right = new tree(7);
    cout << "前序输出：";
    Preorder(root);
    cout << endl << "中序输出：";
    Inorder(root);
    cout << endl << "后序输出：";
    Postorder(root);
    cout << endl;
    node* tree;
    cout << "请按照先序遍历建立数组：";
    creatprethread(tree);
    prevoder(tree);
    return 0;
}
```

输入样例:
见上述代码主函数
即（前序输出为数字顺序）

```
    1
  2   5
3  4 6  7
```

运行结果:



实现操作
链式树
线索树结点类
先序线索化（分为主函数和设置左右标识函数）
先序遍历

分析

本代码实现的是正序先序输出，而以最后一个结点为首结点，逐个访问前驱结点，即可实现逆序输出。更改输出次序，可以实现后序输出。