//5.2.4节 算法NPO

```
//非递归后根遍历算法中的辅助结点类
template<class T>
class AssBinTreeNode
{
private:
        BinTreeNode<T> * ptr    ;
        int flag;
public:
        AssBinTreeNode( BinTreeNode<T> *p=NULL , int i=0 ): ptr(p),flag(i){}   //  构造函数
        int GetFlag() {    return      flag; }
        void SetFlag(int i) { flag = i ; }
        BinTreeNode<T> * GetPtr() {    return ptr; }
        void SetPtr(BinTreeNode<T> * p) { ptr =p ; }
 } ;
//非递归后根遍历以t为根指针的二叉树
template<class T>
void BinTree<T>:: NorecPostOrder( BinTreeNode<T> *t ) const
{
        if ( t==NULL) return;
        AStack<AssBinTreeNode<T>*> s;                          //建立辅助堆栈来记忆访问路径
        AssBinTreeNode<T>* ass,*lass,*rass;
        ass=new AssBinTreeNode<T>();
        ass->SetPtr(t);      ass->SetFlag(0);
        s.Push( ass) ;
        int i=0;
        while (!s.IsEmpty())
        {
                s.Pop(ass);
                t=ass->GetPtr();      i=ass->GetFlag();
                if (i==0)
                {
                        ass->SetFlag(1);
                        s.Push(ass) ;
                        if((t->GetLeft()) !=NULL)
                        {
                                lass=new AssBinTreeNode<T>(t->GetLeft() ,0);
                                lass->SetFlag(0);
                                s.Push( lass) ;
                        }
                }
                if (i==1)
                {
                        ass->SetFlag(2);
                        s.Push( ass) ;
                        if((t->GetRight())!=NULL)
                        {
                                rass=new AssBinTreeNode<T>(t->GetRight(),0);
                                rass->SetFlag(0);
                                s.Push( rass) ;
                        }
                }
                if (i==2)
                        cout<< t->GetData()<<endl;
        }
};
```