排序（1）

- 洗牌算法
  - (2*i)%(n+1)
- 二叉树遍历
  - 前序遍历
    - 递归
      -
```
1 void preOrder1(BinTree *root)      //递归前序遍历
2 {
3     if(root!=NULL)
4     {
5         cout<<root->data<<" ";
6         preOrder1(root->lchild);
7         preOrder1(root->rchild);
8     }
9 }
```
    - 非递归
      -
```
void preOrder2(BinTree *root)      //非递归前序遍历
{
    stack<BinTree*> s;
    BinTree *p=root;
    while(p!=NULL||!s.empty())
    {
        while(p!=NULL)
        {
            cout<<p->data<<" ";
            s.push(p);
            p=p->lchild;
        }
        if(!s.empty())
        {
            p=s.top();
            s.pop();
            p=p->rchild;
        }
    }
}
```
  - 中序遍历
    - 递归
      -
```
void inOrder1(BinTree *root)       //递归中序遍历
{
    if(root!=NULL)
    {
        inOrder1(root->lchild);
        cout<<root->data<<" ";
        inOrder1(root->rchild);
    }
}
```

- 非递归
  -

```
void inOrder2(BinTree *root)       //非递归中序遍历
{
    stack<BinTree*> s;
    BinTree *p=root;
    while(p!=NULL||!s.empty())
    {
        while(p!=NULL)
        {
            s.push(p);
            p=p->lchild;
        }
        if(!s.empty())
        {
            p=s.top();
            cout<<p->data<<" ";
            s.pop();
            p=p->rchild;
        }
    }
}
```

- 后序遍历
  - 递归
    -

```
void postOrder1(BinTree *root)     //递归后序遍历
{
    if(root!=NULL)
    {
        postOrder1(root->lchild);
        postOrder1(root->rchild);
        cout<<root->data<<" ";
    }
}
```

  - 非递归
    -

```
void postOrder2(BinTree *root)     //非递归后序遍历
{
    stack<BTNode*> s;
    BinTree *p=root;
    BTNode *temp;
    while(p!=NULL||!s.empty())
    {
        while(p!=NULL)              //沿左子树一直往下搜索，直至出现没有左子树的结点
        {
            BTNode *btn=(BTNode *)malloc(sizeof(BTNode));
            btn->btnode=p;
            btn->isFirst=true;
            s.push(btn);
            p=p->lchild;
        }
        if(!s.empty())
        {
            temp=s.top();
            s.pop();
            if(temp->isFirst==true)     //表示是第一次出现在栈顶
            {
                temp->isFirst=false;
                s.push(temp);
                p=temp->btnode->rchild;
            }
            else                        //第二次出现在栈顶
            {
                cout<<temp->btnode->data<<" ";
                p=NULL;
            }
        }
    }
}
```

-

幕布 - 思维概要整理工具