带头尾哨兵的双向链表的稳定的插入序算法 解题report

2018011365 张鹤潇 计84

解题思路

Student结构储存学生信息。

linknode结构包含了指向双链表后继、前驱和对应Student的三个指针。

initlinklist函数生成头哨兵和尾哨兵，并在双链表中插入节点。

Insertsort函数的实现是整个程序最大的难点。我定义了向下遍历的指针TravDown和向上遍历的指针TravUp。为了简化边界条件判断，引入标记TravDown节点位置的j；对于j号结点TravDown，对用已经排好序的序列[1,j-1]向上遍历，找到最靠前的group值比Travdown小的结点Travup，将Travdown插到Travup后面。如果这样的Travup不存在，则将Travdown插到头哨兵的后面。

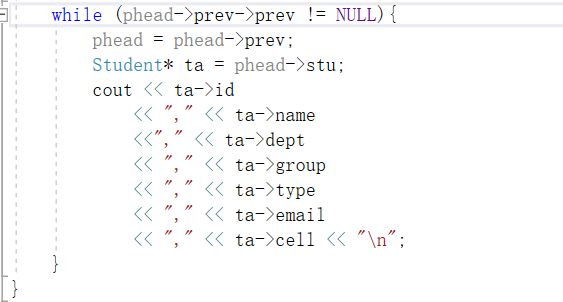
readStructsIntoList函数将输入的数据读入链表中。

writeListIntoStructs函数顺序输出结点信息。（以上两个函数出于个人偏好使用了递归）

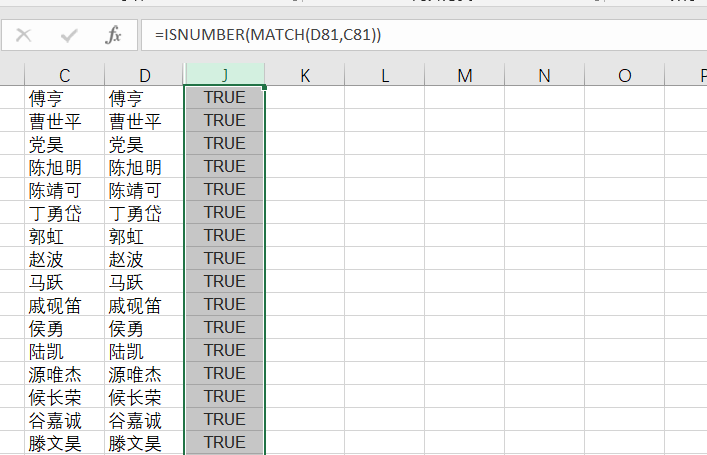
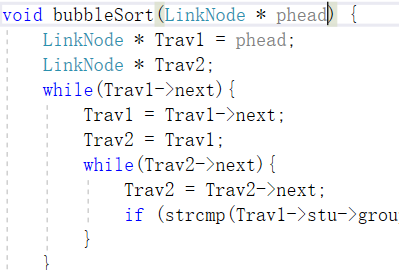
releaseLinkList函数在输出结束后释放内存。

测试排序后双向链表的完整性和排序算法的稳定性

为了便于完成测试，将处理结果和原始数据储存为csv格式，利用excel进行比较。

1. 将输出函数略作修改，将排序后的双链表反向输出，检查结果，所有数据正确地逆序输出。

1. 将排序后的结果与稳定的冒泡排序以及excel排序得到的结果进行比较，二者完全相同。可见排序算法稳定。



1. 将排序指标改为院系、姓名，①②测试仍然成功。

双向连接和头尾哨兵的作用

在本程序中，双向连接使排序时对双链表上下进行遍历比较方便。

头尾哨兵的存在有助于避免排序/输出时访问到空指针导致程序崩溃。