

高级程序设计实践

实验报告

|  |  |
| --- | --- |
| 题 目： | **实验报告** |
| 学生姓名： | **周天赐** |
| 指导教师： | **李玺** |
| 学 院： | **计算机学院** |
| 专业班级： | **计科1803班** |

本科生院制

2019年12月

## 一、题目要求

**单链表、队列、二叉树查找和排序算法的实践**

实践目的或任务：通过指导学生上机实践，结合航空机票订票系统，对单链表、队列、二叉树查找和排序算法的基本概念及其不同的实现方法的理论得到进一步的掌握，并对在不同存储结构上实现不同的运算方式和技巧有所体会。

**实践基本要求：**

1、了解实验目的及实验原理；

2、编写程序，并附上程序代码和结果图；

3、总结在编程过程中遇到的问题、解决办法和收获。

**实践的内容或要求：**

1、基本操作（必做部分）

（1）编写函数，实现输入一组元素，建立一个带头结点的单链表；对该链表进行非递减排序；实现在非递减有序链表中删除值为x的结点；

（2）编写函数，采用链式存储和顺序存储实现队列的初始化、入队、出队操作；

（3）编写函数，建立有序表，利用二叉排序树的插入算法建立二叉排序树；在以上二叉排序树中删除某一指定关键字元素；采用折半查找实现某一已知的关键字的查找(采用顺序表存储结构)

（4）选用1-3的数据结构，编写程序实现下述五种算法：简单插入排序，冒泡排序，快速排序，归并排序，堆排序。

2、航空机票订票系统设计（四选一选做部分）

（1）设计每条航线所涉及的信息，如终点站名、航班号、飞机号、飞机周日（星期几）、乘员定额、余票量、订定票的客户名单（包括姓名、订票量、舱位等级1，2或3）等；

（2）结合基本操作的单链表、队列、二叉树等数据结构以及排序算法，设计机票系统的查询、订票、退票等功能；

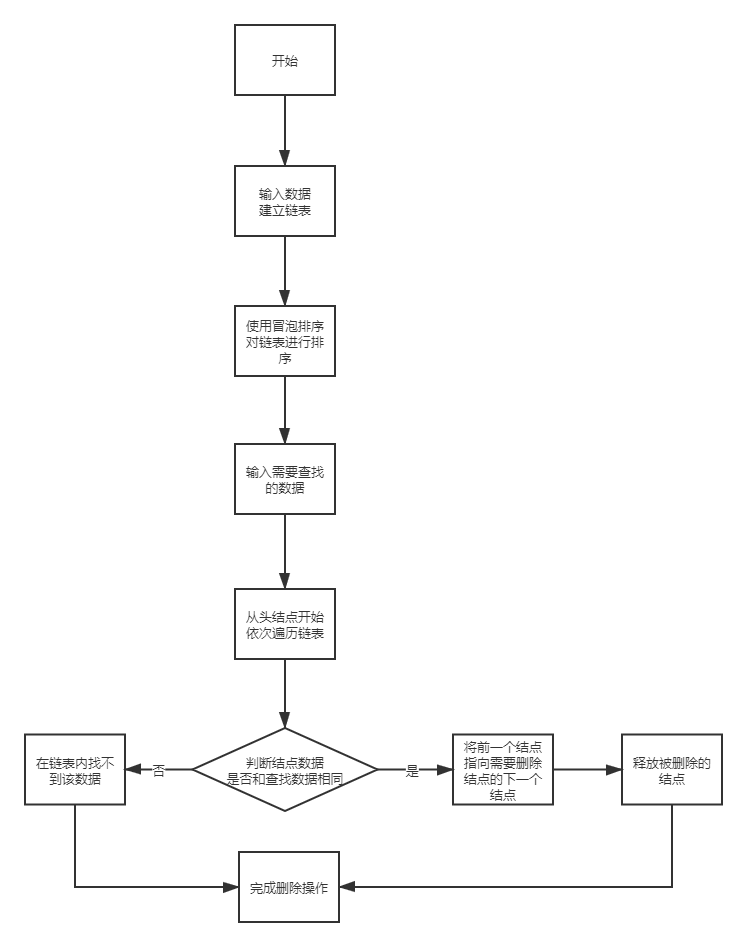
（3）设计并实现人机交互友好的界面或菜单。

## 二、设计思路及实现过程

**基本操作部分：**

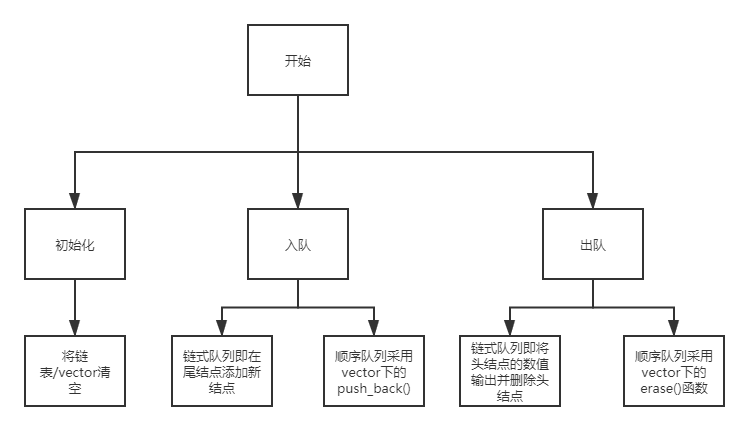
主要考察的是数据结构相关的知识，其中包括链表、队列、二叉树、排序。

1. 链表



首先创建一个结点类，其中成员变量包括结点储存的数据以及结点直接指向的下一个结点地址。将链表和其相关操作封装为一个类，在创建链表时，即将多个结点连接，其中最后一个尾结点指向的地址设为空，即生成了一个链表。排序时从头结点开始遍历，利用冒泡排序的方式，将链表排序。删除操作同样是从头结点开始遍历，进行数据匹配，如果寻找到含有需要删除的数据的结点那么就将前一个结点指向需要删除结点的下一个结点，再释放被删除的结点，即可完成删除操作。

1. 队列



这里要求使用链式存储和顺序存储两种方式实现队列。

对于链式存储，即生成一个链表并对这个链表进行一系列的操作，初始化即将头结点设为空并且如果存在其他结点就将其他结点释放，入队对应的是在链表最后增加一个结点，出队则对应删除头结点并将头结点设为原来的下一个结点。

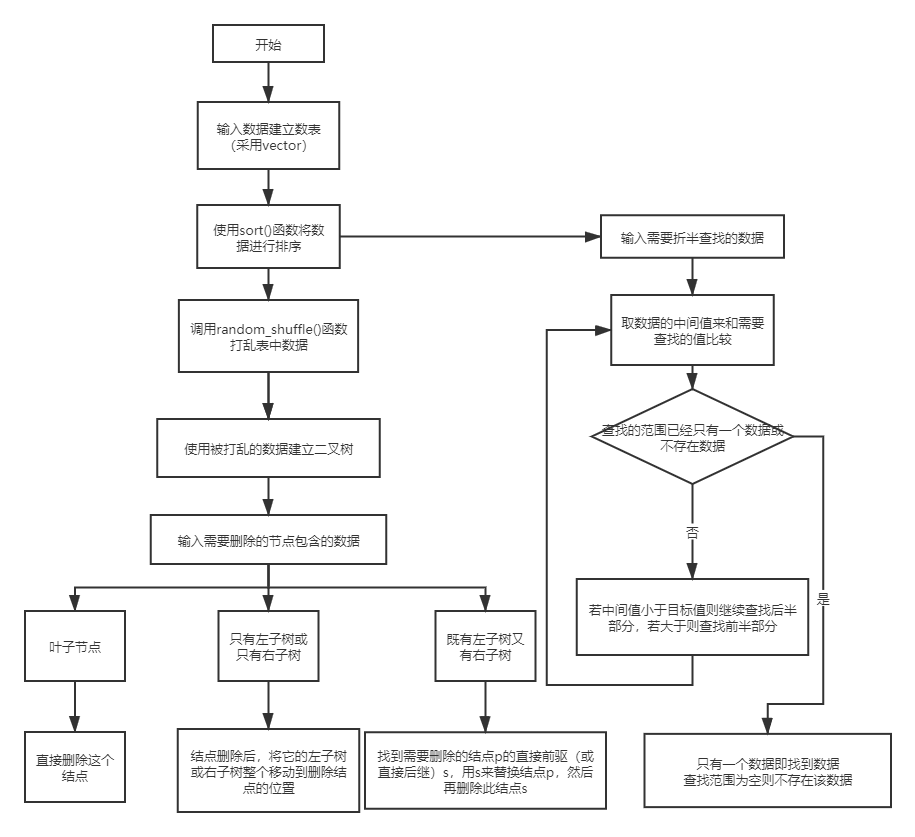
对于顺序存储，在这里使用的是vector类来实现队列，初始化可以直接重新生成一个新的vector，入队对应的是push\_back()，出队则对应erase()，直接删除第一个数据即可。

1. 二叉树和折半查找

建立有序表这里采用的是vector来存放数据并用sort()函数来对其排序，排序完成后直接调用random\_shuffle()函数打乱表中数据来建立二叉树。二叉树的建立需要新建一个二叉树结点类，其中包括了其储存的数据和指向的左右子结点地址。

对于二叉树删除指定结点，这里分为了几种情况：第一种情况是需要删除的结点是叶子节点即没有左右子树，那么直接删除这个结点即可，因为这个结点的删除没有对二叉树本身产生影响；第二种情况是要删除的结点只有左子树或只有右子树，结点删除后，将它的左子树或右子树整个移动到删除结点的位置即可；第三种情况是要删除的结点既有左子树又有右子树，那么只要找到需要删除的结点p的直接前驱（或直接后继）s，用s来替换结点p，然后再删除此结点s。

折半查找即对于有序的数据，每次取数据的中间值来和需要查找的值比较，若中间值小于目标值则继续查找后半部分，若大于则查找前半部分，一直循环到查找的范围已经只有一个数据或不存在数据，即对应了找到数据和不存在该数据两种情况。



1. 排序

简单插入排序：

其主要思想就是逐个元素找到其应该呆在的地方。

（1）取出第一个元素；

（2）取出第二个元素，与第一个元素比较，小于第一个元素则排在第一个元素前面，大于则排在后面；

（3）取出第n个元素，在前n-1个已排序好的元素中从后往前找到能依旧保持有序的位置插入。

冒泡排序：

其主要思想就是不断遍历，两两之间进行比较，并按大小顺序进行交换，则每次最大的数据都会到最尾部。

对有n个数据的数组进行n次遍历，每次遍历到第n+1-i个数据，如果发现每相邻两个数据之间不满足左边小于右边的情况，就交换数据，那么在n次遍历后，就完成了升序冒泡排序。

快速排序：

快速排序使用分治的思想，通过一趟排序将待排序列分割成两部分，其中一部分记录的关键字均比另一部分记录的关键字小。之后分别对这两部分记录继续进行排序，以达到整个序列有序的目的。

(1)选择基准：在待排序列中，按照某种方式挑出一个元素，作为基准；

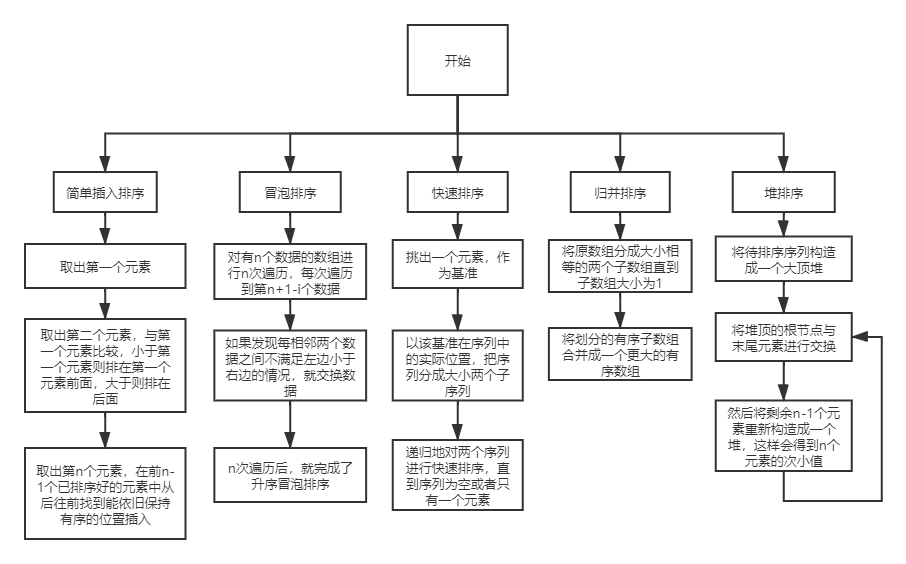
(2)分割操作：以该基准在序列中的实际位置，把序列分成两个子序列。此时，在基准左边的元素都比该基准小，在基准右边的元素都比基准大；

(3)递归地对两个序列进行快速排序，直到序列为空或者只有一个元素。

归并排序：

归并排序基于分治的递归算法。它不断地将原数组分成大小相等的两个子数组(或者是相差1的两个数组)，最终当划分的子数组大小为1时，将划分的有序子数组合并成一个更大的有序数组。

堆排序：

将待排序序列构造成一个大顶堆，此时，整个序列的最大值就是堆顶的根节点。将其与末尾元素进行交换，此时末尾就为最大值。然后将剩余n-1个元素重新构造成一个堆，这样会得到n个元素的次小值。如此反复执行，便能得到一个有序序列了。

**航空机票订票系统设计：**

1. 航班信息的设计：

考虑到航班信息直接预设在代码中可维护性较低，故将航班信息放在一个文档中，程序运行时读取文档中的信息来达到设定航班的目的，这样直接修改文档即可修改航班信息。如果进一步完善程序可以加入服务器端，服务器端可以将航班信息传到客户端，并录入该文档，较为方便。

设定一个航班类，在运行时将文档内的航班全部实例化。

1. 客户信息的设计：

客户设为一个类，在使用时实例化，在每一个航班对象中有一个客户类型的Vector用来储存该次航班的所有客户信息。

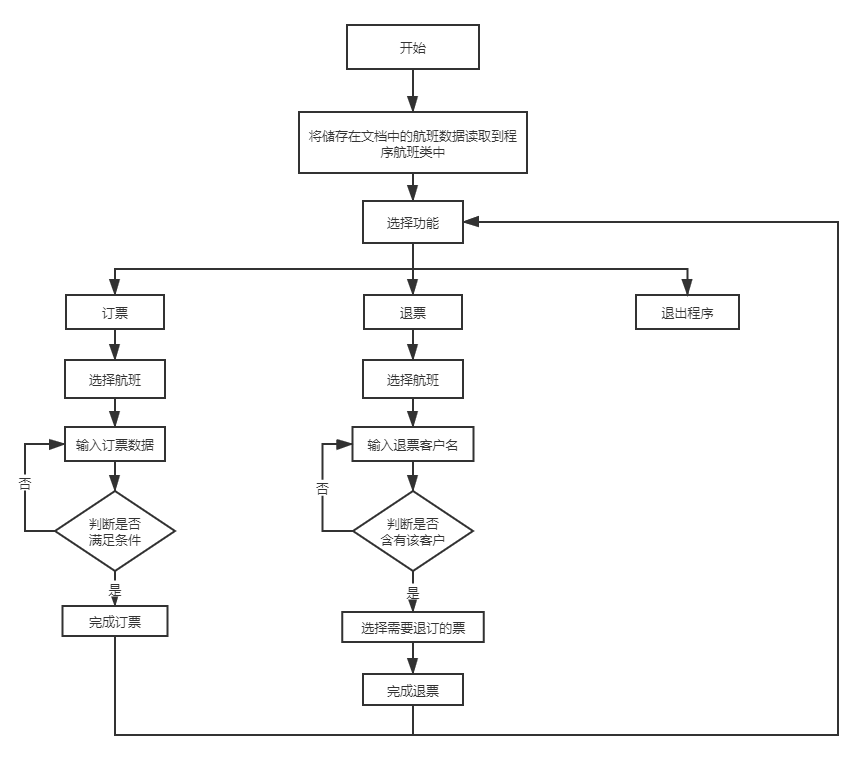
1. 订票：

订票的原理很简单，判断余票量和需要订票的量的关系，满足订票条件时实例化顾客类并且更新航班信息。在这里会判断多次订票时客户是否已经订票，如果已经订票则直接更新该顾客的顾客对象即可。

1. 退票：

退票时要求顾客选择航班，并且输入顾客信息，如果有订票信息即可进行退票，退票需要更新顾客信息和航班信息，在退票完成后会执行清理工作，清理并没有订购任何一张票的顾客。

1. 程序结束：

在程序结束时，会将航班信息和顾客信息写入文档中，可在程序运行结束以后再查看航班情况和顾客的订票信息。

## 三、难点及解决

本次实验遇到的难点主要有两个：

1. 五种排序算法：

这五种排序在之前的数据结构课程中已经学过了大部分，但也只是基于理论基础的学过，在实际实现的过程中，几种需要利用递归的排序还是存在着一定难度的。

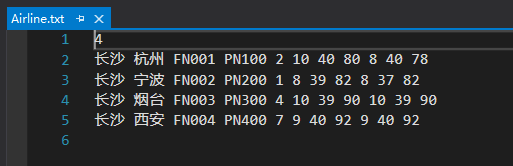
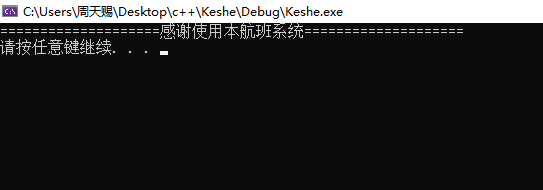
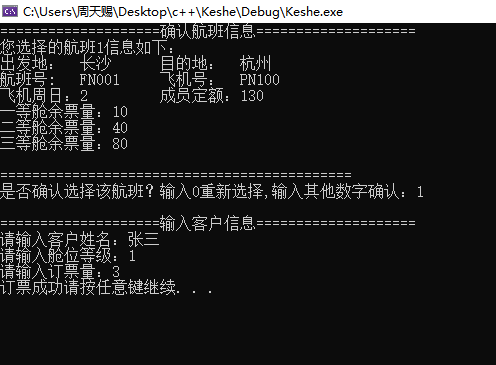
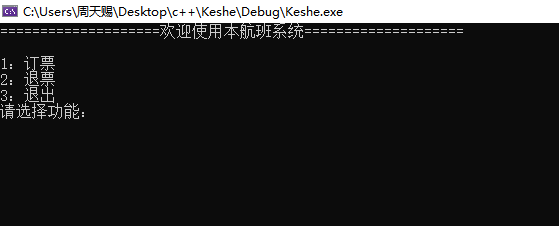
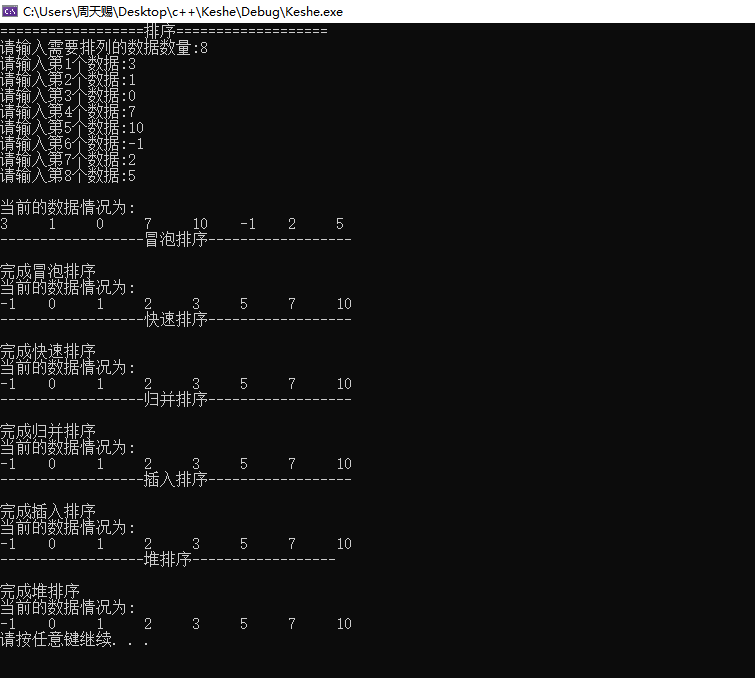
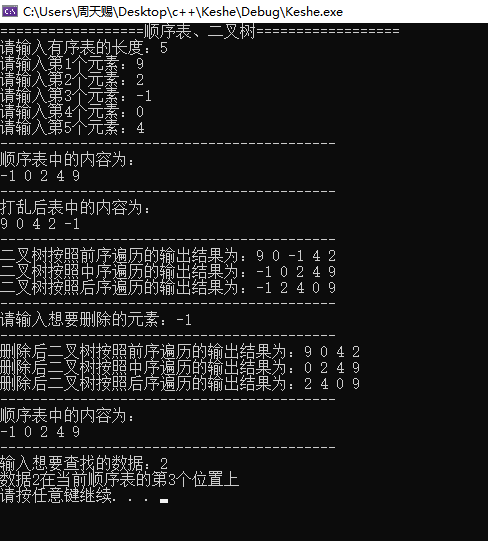
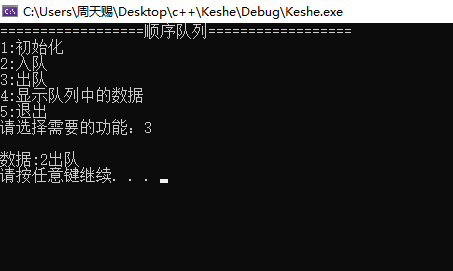
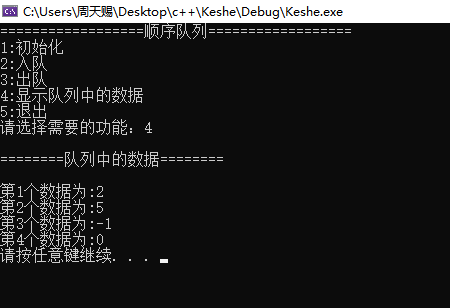
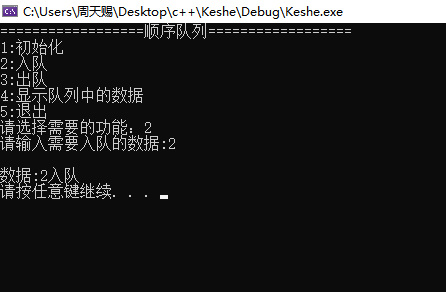
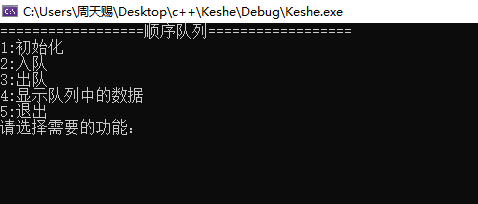
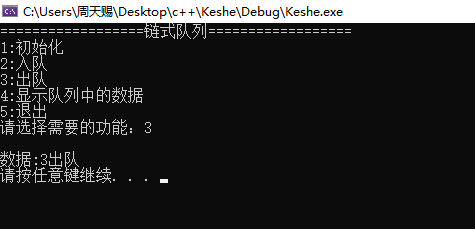
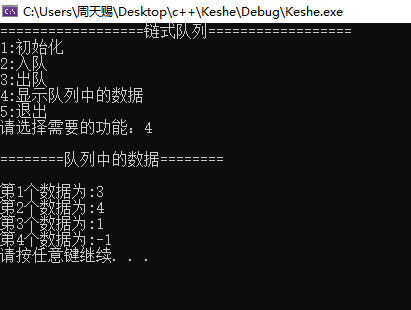
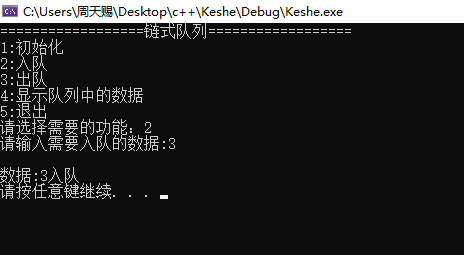
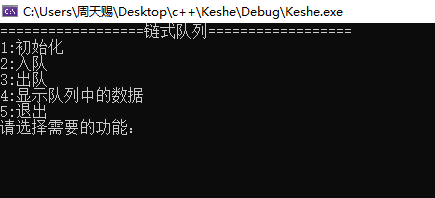
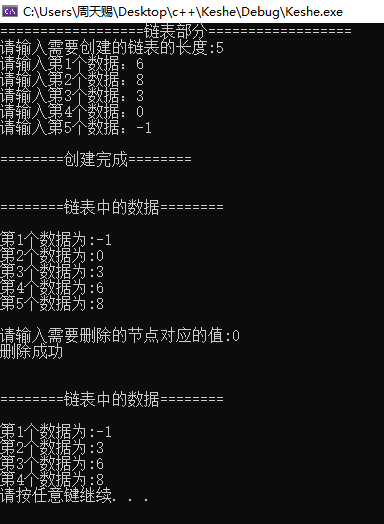
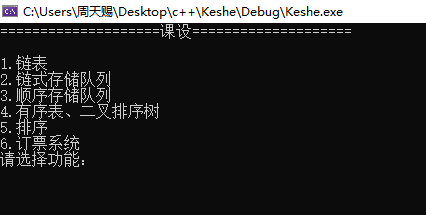
在自己多次尝试后始终没有办法达到理想的效果，最终选择查阅数据结构的书籍和网上的排序相关资料，完成了这五种排序的实现。

1. 机票订票系统：

在订票系统中，航班信息的储存一开始是我遇到的比较头疼的问题，如果直接将其预设在代码中，修改起来比较麻烦。想到在之前的学生缴费信息的上机实践，转用文本文档按照格式储存航班信息，然后在代码中加入加载信息的相关代码，这样需要修改航班的信息时只需要修改文本文档内的内容即可。

在之后遇到的问题则是顾客订票的问题，是将顾客单独处理还是在航班内加入顾客，是我当时比较纠结的问题，然后发现如果顾客单独处理，会导致退票查询较为复杂，所以转为在每个航班内加入一个顾客类数组，每次订票都是针对特定航班订票。

## 运行及测试结果



## 心得体会

本次高级程序设计完成了数据结构中部分的实现和使用，且完成了一个航班订票系统，内容多样而具有较强的实际性。

通过这次高级程序设计实践，让我对于一些常见常用的数据结构的认识更加深入，同时本次实践基于C++完成，对于C++的抽象、封装、继承、多态的特性有了切实的体会，对于其本身编程语言的掌握也有所提高。

在实践的过程中也会遇到一系列的问题，在现在信息多样化的时代，解决问题的途径很多，翻阅书籍，查阅别人的心得，都是可行的方案，在这次实践中我也遇到了一些比较棘手的问题，在查询了相关的资料以后还是能够较好地解决的。

最后设计的航班订票系统虽然界面比较简易，但是其功能还是较为完整的，不过还是有较多可以完善的地方，比如界面就可以做的更加美观一些，同时航班的数量比较少，可以多增加一些航班，同时加入查询从某地到某地是否有航班这类的功能，让这个系统更加贴近日常使用，优化其人机交互体验。

总体而言，这次高级程序设计实践让我受益匪浅。