**实 验 报 告**

|  |  |
| --- | --- |
| **课程名称：** | **数据结构与算法分析** |
| **学生姓名：** | **叶子繁** |
| **学生学号：** | **201730684427** |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2018-2019第一学期** |

**软件学院**

**2018年10月**

# 线性表的实现及应用

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **地 点：** | | | B7 楼 | | 233 房； | | **实验台号：** | | 233-87 |
| **实验日期与时间：** | | | 2018.10.12 14：00 | | | | **评 分：** | |  |
| **预习检查纪录：** | | |  | | | | **实验教师：** | |  |
| **电子文档存放位置：** | | | /home/pc/下载/exp 1-1/ | | | | | | |
| **电子文档文件名：** | | | Llist.h Alist.h testAlist.cpp testLlist.cpp main.cpp final.exe | | | | | | |
| **批改意见：** | | |  | | | | | | |
| 出勤情况 | 实验报告质量 | | | | | |
| 任务完成情况 | | 源代码注释 | | 报告内容 | |
| □ 出勤  □ 缺勤  □ 早退  □ 请假 | □ 完成  □ 未完成 | | □ 优秀  □ 良好  □ 通过  □ 未通过 | | □ 优秀  □ 良好  □ 通过  □ 未通过 | |

# 报告内容

1、实验目标

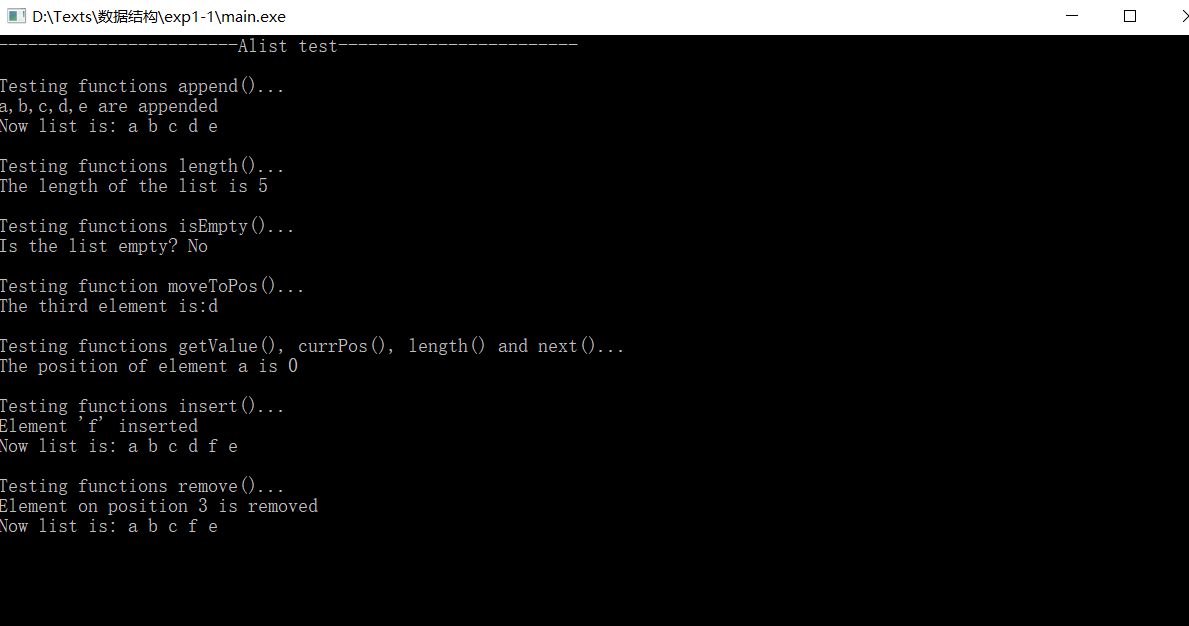
2、实验环境

系统：Windows 10 & Ubuntu 16.04

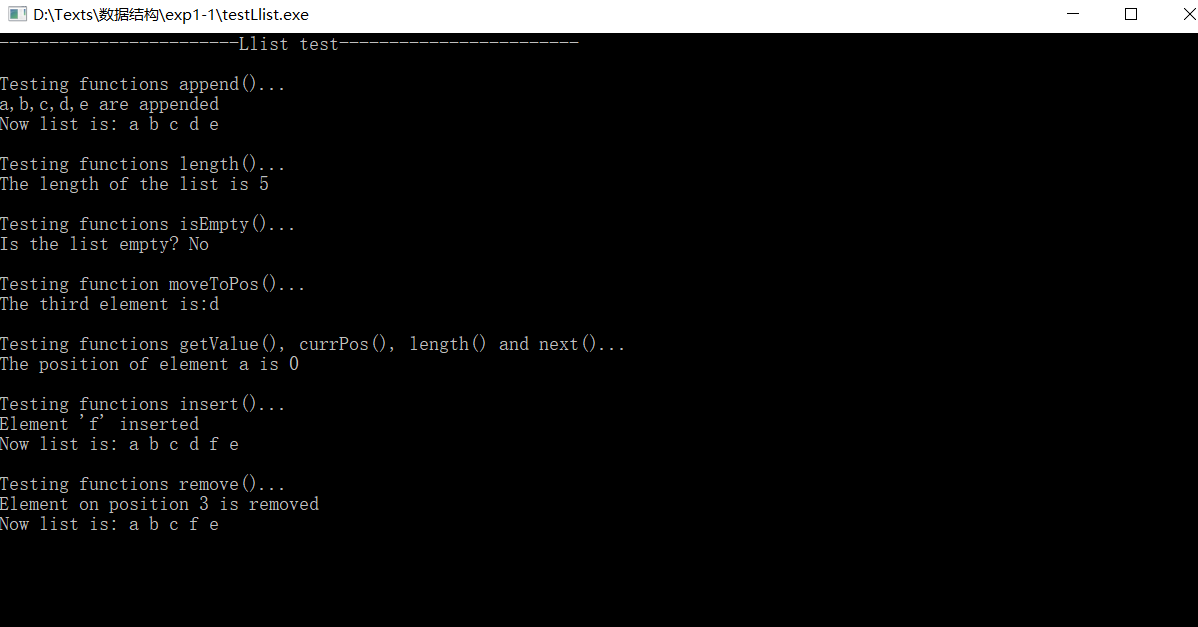
IDE：Visual Studio Code & Code::Blocks

3、线性表的操作实现

Alist的测试截图



Llist的测试截图



4、List数据结构的应用：汽车管理系统

4.1 关键数据结构

Llist链表

4. 2算法分析（空间、时间）

排序并插入：O(n);

删除：O(n)

显示某一记录：O(n)

显示全部记录：O(n)

删除全部记录：O(n)

修改某一记录：O(n)

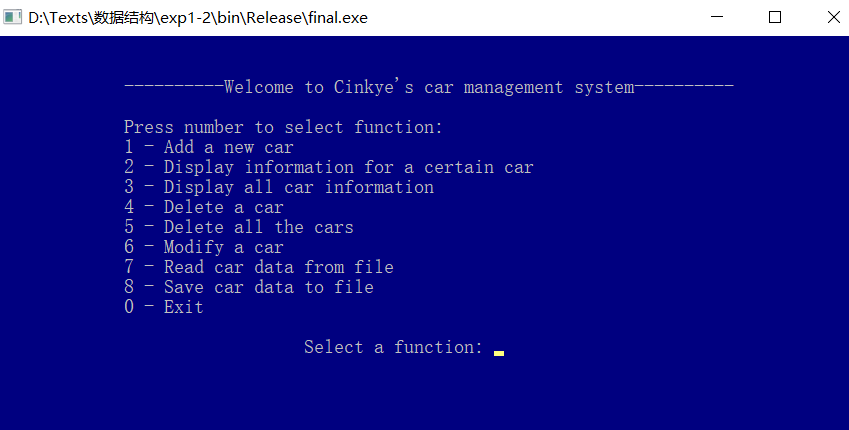
文件读取/输出：O(n)

4. 3 错误处理

对用户输入的所有数据做有效性检测，如果输入无效，则回到主界面或者提示重新输入

4.4 运行结果

主界面：

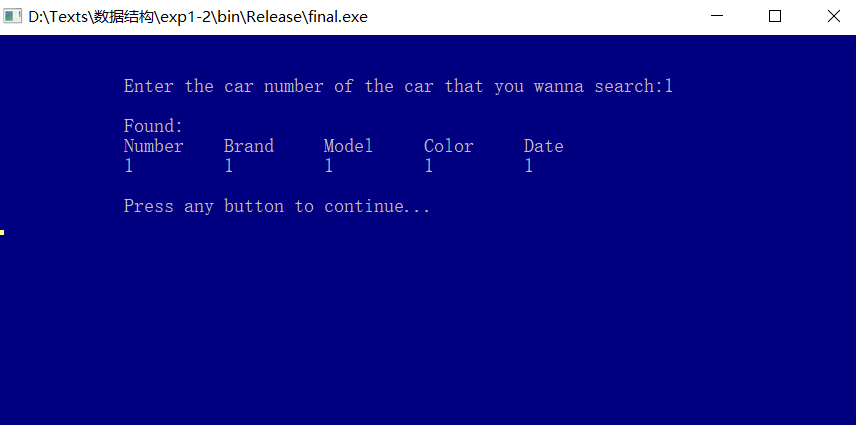


添加信息：

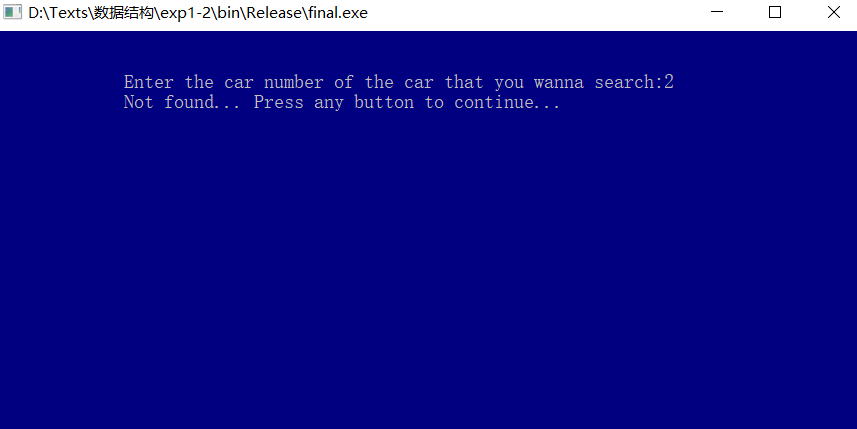


查看某一信息（以车牌为key）：

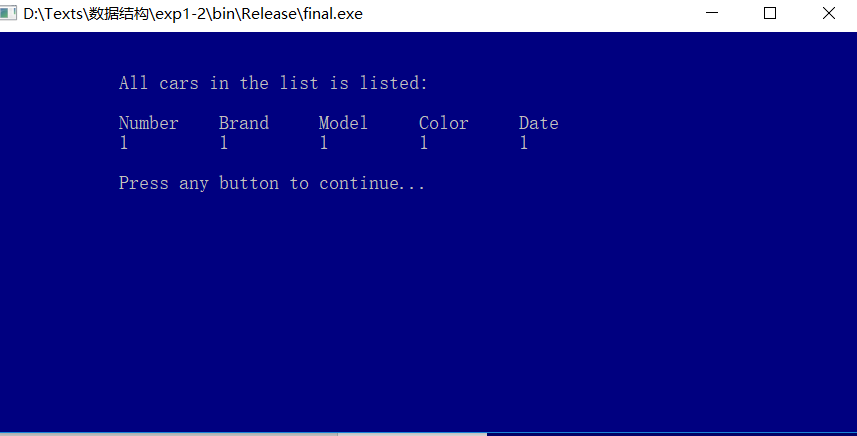
找到：



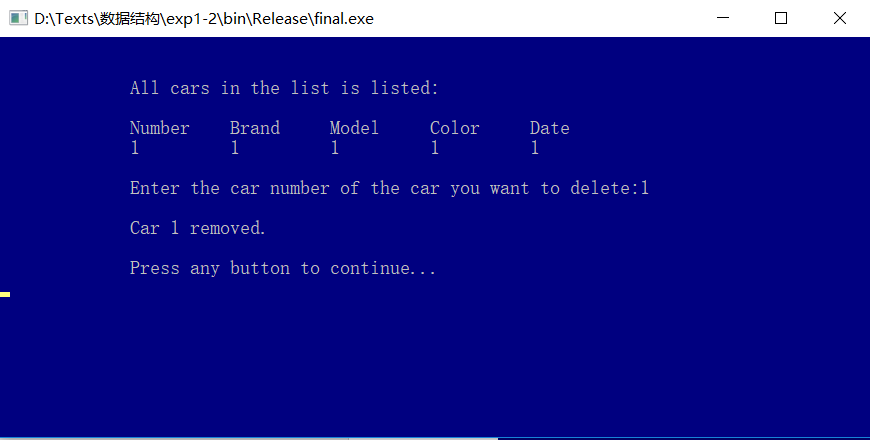
没找到：



查看所有记录：



删除特定记录：

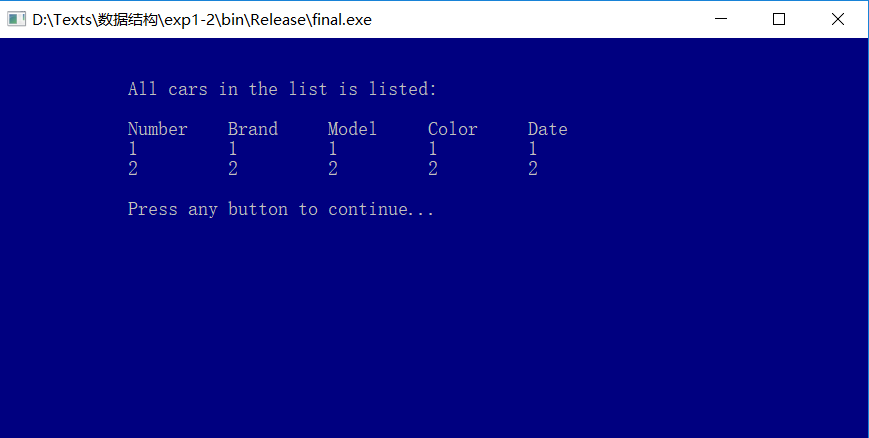


从文件中读取：

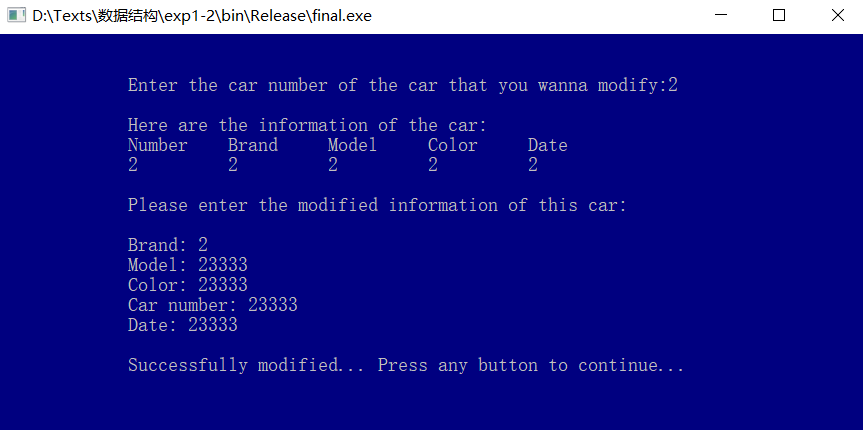


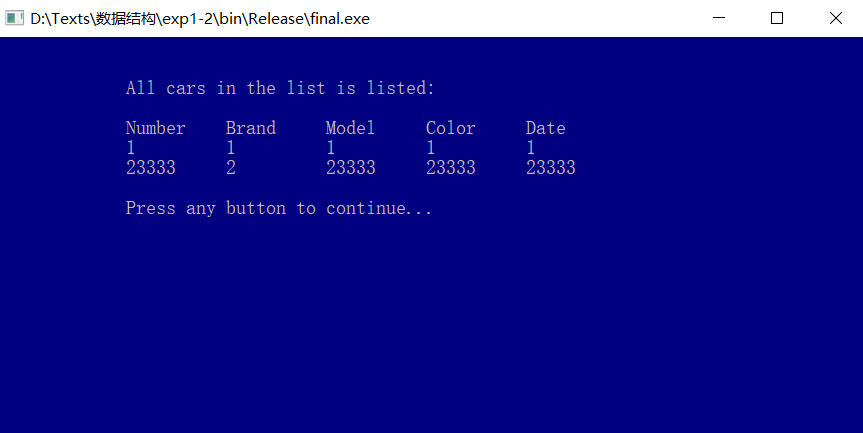


查看所有记录：

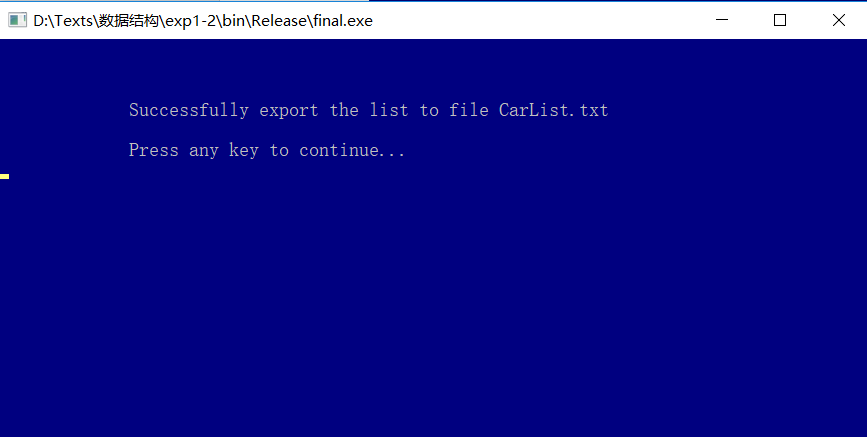


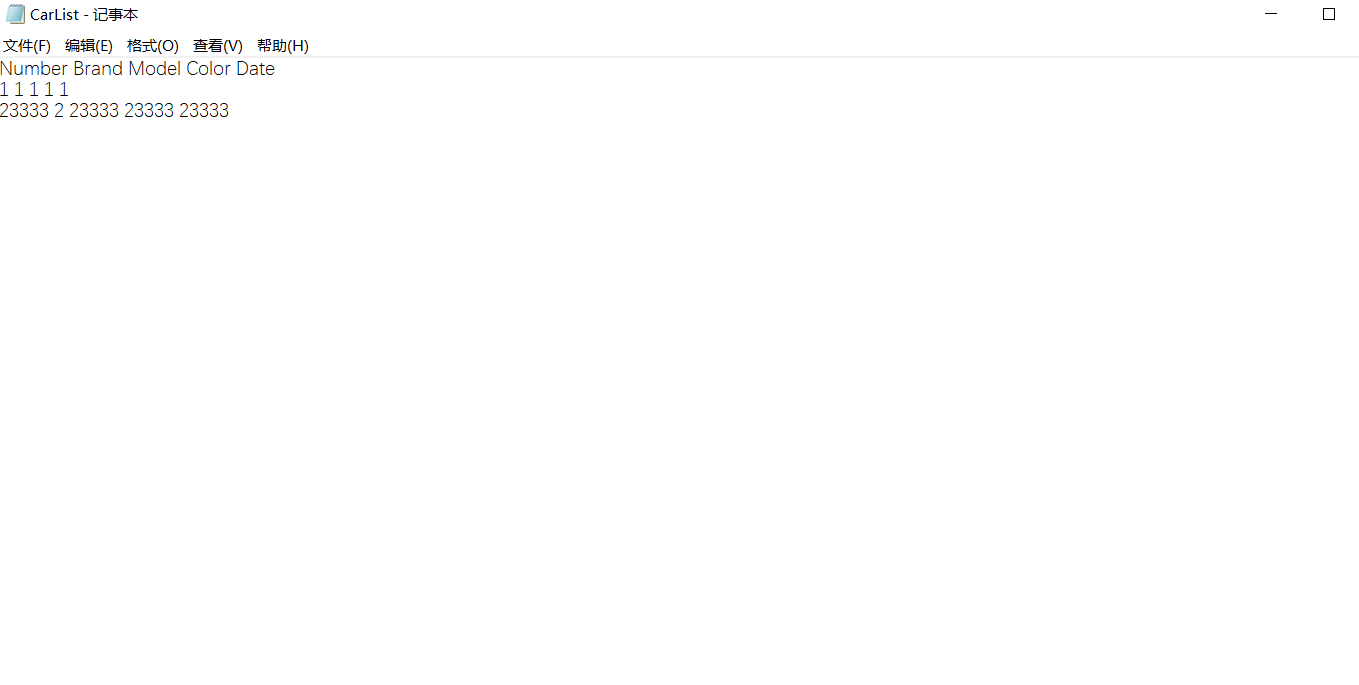
修改特定记录：





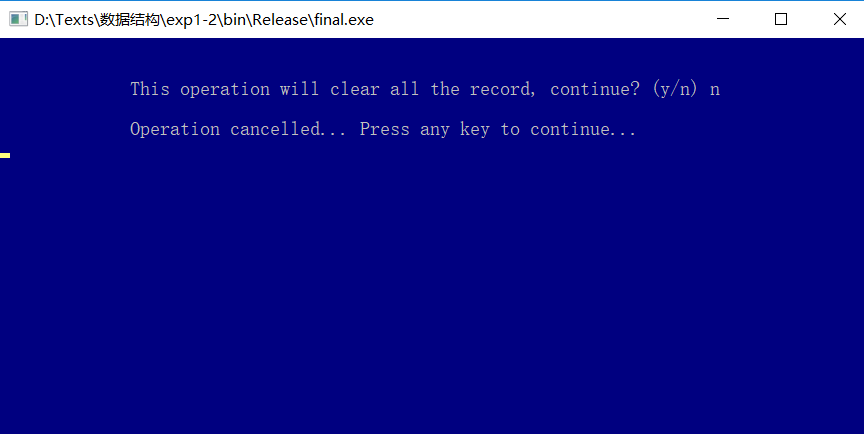
输出到文件：





删除所有记录：

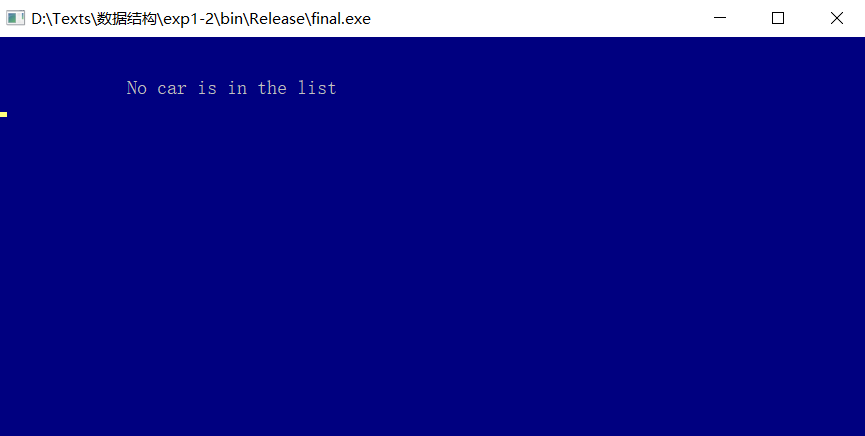
取消删除：



确认删除：



查看所有记录：



5、总结

　边界的检查以及输入的控制十分重要，否则越界访问会带来意想不到的效果，实现 Llist.h 和 Alist.h 看起来并不困难，但是在使用自己写的头文件进行编程的时候发生了许多边界没有控制好的状况。只有好的类的实现才能在应用类时少一点错误，但也只有在应用时才能发现类的实现中一些不容易找到的错误。