

C++程序设计课程设计

课设报告

**2024/2025(2)**



实验题目 物业维修管理系统

学生姓名 蒋徐杰

学生学号 302024315094

学生班级 软件工程4班

任课教师 廖锋峰

提交日期 2025.5.29

**计算机科学与技术学院**

**物业维修管理系统 实验报告**

1. **实验内容**

本系统实现物业维修管理功能，核心模块包括：

* **维修记录管理**：支持记录的增、删、改、查，记录包含住户信息、预约 / 维修日期、维修内容、费用等。
* **日期处理**：自定义Date类实现日期的格式化输出、输入、比较及当前日期获取。
* **数据存储**：使用文件存储维修记录，支持数据的加载与保存。
* **统计与排序**：按楼号统计维修费用，按实际维修日期排序记录。

1. **运行环境**

物业维修管理系统 在CLion 2025.1平台下开发，操作系统：Windows 11 23H2。

硬件环境：

处理器：Intel(R) Core(TM) i9-14900HX 2.20 GHz

内存：32.0 GB

系统类型：64 位操作系统, 基于 x64 的处理器

1. **实验课题分析**

**3.1主要功能：**该物业维修管理系统集记录操作、数据存储与检查于一体。可实现维修记录增删改、预约到期及精确 / 模糊查询、按楼号统计金额数量、按实际日期浏览记录，还能在文件存储与加载数据，并进行合法性检查。-

**记录操作功能：**

1. **增删改功能**：可增加、删除维修记录，还能修改已有的维修记录 。
2. **查询功能**
   * **预约到期查询**：查询预约到期的维修记录。
   * **精确 / 模糊查询**：对维修记录进行精确或模糊查询。
3. **统计功能**：依据楼号统计每幢楼的维修记录数量，以及总维修收费金额、总维修材料成本金额。
4. **浏览功能**：按实际维修日期排序，浏览所有维修记录。

**数据存储功能：**

能将所有维修记录保存到文件，并能从文件加载相关信息 。

**数据检查功能：**

对各项数据进行合法性检查 。

**3.2 系统分析及设计**

**核心功能：维修记录管理**

**1. 记录增删改操作**

* **新增记录（add()）**
  + 用户输入住户名、预约日期、维修日期、维修内容、收费金额、材料成本、维修人、备注（格式：YYYY-MM-DD），系统创建RepairRecord对象并添加到链表末尾。
  + 支持直接传入已有记录对象批量添加（add(const RepairRecord&)）。
* **删除记录（del()）**
  + 用户输入记录序号，系统通过序号定位节点并删除。
  + 处理头节点、尾节点及中间节点的删除逻辑，更新链表长度n。
* **修改记录（upgrade()）**
  + 用户选择需修改的字段（如住户、日期、金额等），系统交互式更新对应属性。
  + 支持日期格式校验（需符合YYYY-MM-DD）和数值输入验证。

**2. 记录查询功能**

* **预约到期查询（findDeadRecord()）**
  + 对比当前日期（通过Date::getCurrentDate()获取）与维修记录的实际日期，筛选出实际日期早于当前日期的记录并显示。
* **模糊 / 精确查询（query()）**
  + 用户输入关键词，系统在维修记录的住户名、维修内容、日期、金额、维修人、备注等字段中匹配，支持部分匹配（如输入 “水管” 可匹配 “水管爆裂维修”）。
  + 使用RepairRecord::find(str)方法逐字段检索，返回所有包含关键词的记录。

**3. 记录统计功能（stats()）**

* 按住户名中的楼号（如 “1 号楼 - 101” 提取 “1” 作为楼号）分组，统计各楼的总收费金额、总材料成本。
* 使用vector存储各楼号的统计结果，遍历链表累加数据，最终以表格形式展示楼号、总金额和成本。

**4. 记录浏览功能（display()）**

* 按实际维修日期排序显示所有记录（通过sortByDate()实现归并排序）。
* 表格化输出，字段包括序号、住户、预约日期、维修日期、维修内容（自动填充空格保证对齐）、金额（保留两位小数）、维修人、备注。

**数据存储与加载**

**1. 文件存储（save()）**

* 将链表中的所有记录写入指定文件（默认test.txt）
* 日期自动格式化为YYYY-MM-DD，金额保留原始精度。

**2. 文件加载（load(const string& name)）**

* 读取文件内容，解析为RepairRecord对象并构建链表。
* 若文件不存在，自动创建新文件并初始化记录数为 0。

**日期处理功能（Date类）**

**1. 日期格式化**

* toString()：将日期转为YYYY-MM-DD格式字符串。
* 重载<<运算符：直接输出日期到流（如控制台或文件），自动补全前导零（如5月转为05）。

**2. 当前日期获取（getCurrentDate()）**

* 通过std::time和std::localtime获取系统当前时间，自动设置Year、Month、Day属性。

**3. 日期比较**

* 重载<和>运算符：按年、月、日依次比较，用于记录排序和到期判断（如if\_dead()方法）。

**四、辅助功能**

**1. 数据校验**

* 日期输入时校验格式（通过istream& operator>>解析字符串，提取年 / 月 / 日数值，忽略非法字符）。
* 删除 / 修改记录时校验序号有效性（num < 1 || num > n时提示重新输入）。

**2. 排序功能（sortByDate()）**

* 使用归并排序（merge和sortByDate(Node\*)方法）对链表按实际维修日期升序排列，保证浏览时按时间顺序展示。

**五、典型操作流程示例**

1. **新增记录**输入：1号楼-101 2025-05-10 2025-05-11 水（水管爆裂维修） 300 180 赵六 否  
   系统解析为RepairRecord对象，添加到链表并更新文件。
2. **查询 “水管” 相关记录**调用query("水管")，匹配维修内容包含 “水管” 的记录，输出对应条目。
3. **统计各楼收费**遍历链表，按楼号分组累加Charge和Cost。

**3.3系统的实现**

**（1）类的编写**

系统工程名为：RepairSystem。包含了Date类，RepairRecord类，List类。

具体类结构声明如下：

**Date类：**

class Date{  
private:  
 int Year;  
 int Month;  
 int Day;  
public:  
 Date(const Date&);  
 explicit Date(int y=2025, int m=5, int d=1);  
 string toString() const;  
 void getCurrentDate();  
 void setYear(int y){ Year = y; }  
 void setMonth(int m){ Month = m; }  
 void setDay(int d){ Day = d; }  
 friend ostream& operator<<(ostream&,const Date&);  
 friend istream& operator>>(istream&,Date&);  
 bool operator<(const Date& t) const;  
 bool operator>(const Date& t) const;  
};

**RepaurRecord类**

class RepairRecord{  
private:  
 string Resident; //维修住户  
 Date ReservationDate; //预约日期  
 Date ActualDate; //维修日期  
 string Content; //维修内容  
 double Charge; //收费金额  
 double Cost; //材料成本  
 string RepairMan; //维修人  
 string Note; //备注(是否返修)  
 string date;  
public:  
 RepairRecord();  
 RepairRecord(const RepairRecord&);  
 RepairRecord(const string& rt, const Date& re, const Date& ae,const string& ct, const double& ce, const double& cst, const string& rn, const string& nt):  
 Resident(rt), ReservationDate(re), ActualDate(ae), Content(ct), Charge(ce), Cost(cst), RepairMan(rn), Note(nt){ date = "None"; };  
 void print();  
 void upgrade();  
 Date getActualDate(){ return ActualDate; }  
 string getResident(){ return Resident; }  
 Date getReservationDate(){ return ReservationDate; }  
 string getContent(){ return Content;}  
 double getCharge() const{ return Charge;}  
 double getCost() const{ return Cost;}  
 string getRepairMan(){ return RepairMan;}  
 string getNote(){ return Note;}  
 int getBuildNumber();  
 bool if\_dead();  
 bool find(string);  
 friend ostream& operator<<(ostream&,const RepairRecord&);  
};

**（2）链表的使用**

系统实现采用文件的输入输出流对文本数据进行读取与写入，数据的存储组织使用了单向链表。

因为图书管理系统在查找、修改、添加的时候都需要处理大量的数据，所以使用链表十分必要。在RepairRecord类的基础上定义一个对应的list类来管理维修记录数据，具体的结构声明如下：

**List类**

class list{  
private:  
 Node \*head;  
 Node \*tail;  
 int n;  
 int cnt;  
 string filename;  
public:  
 list();  
 ~list();  
 void add();  
 void add(const RepairRecord& );  
 void del();  
 void upgrade();  
 void display();  
 void findDeadRecord();  
 void stats();  
 void query();  
 void query(string str);  
 void sortByDate();  
 Node\* sortByDate(Node\*);  
 Node\* merge(Node\*, Node\*);  
 void load(const string&);  
 void save();  
 int size() const{ return n; }  
};

在运用时，令当前记录的next结点指向新的记录结点，即结点的指针next保存新的图书结点的地址（如下图1所示），以此类推，所有维修记录信息就通过链表的形式串联起来了。

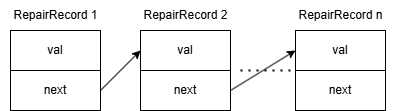


图 1维修记录链表结构示例

**维修记录的查找：**

从第一个节点开始，调用RepairRecord类的find函数判断是否存在要查询的数据，不一致时，p = p->next，处理第二个数据；如果存在，调用函数p->val.print()输出关于该维修记录的所有信息。

如果没有找到可以匹配的，则cout << "未找到!!!" << endl;并结束循环。

**维修记录查找的流程图如下：**

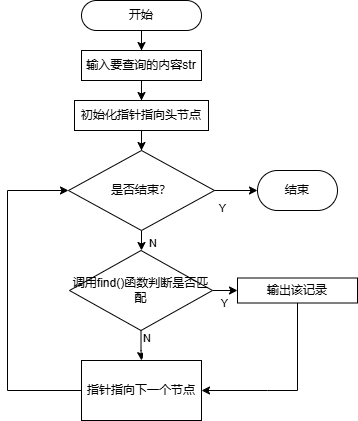
****

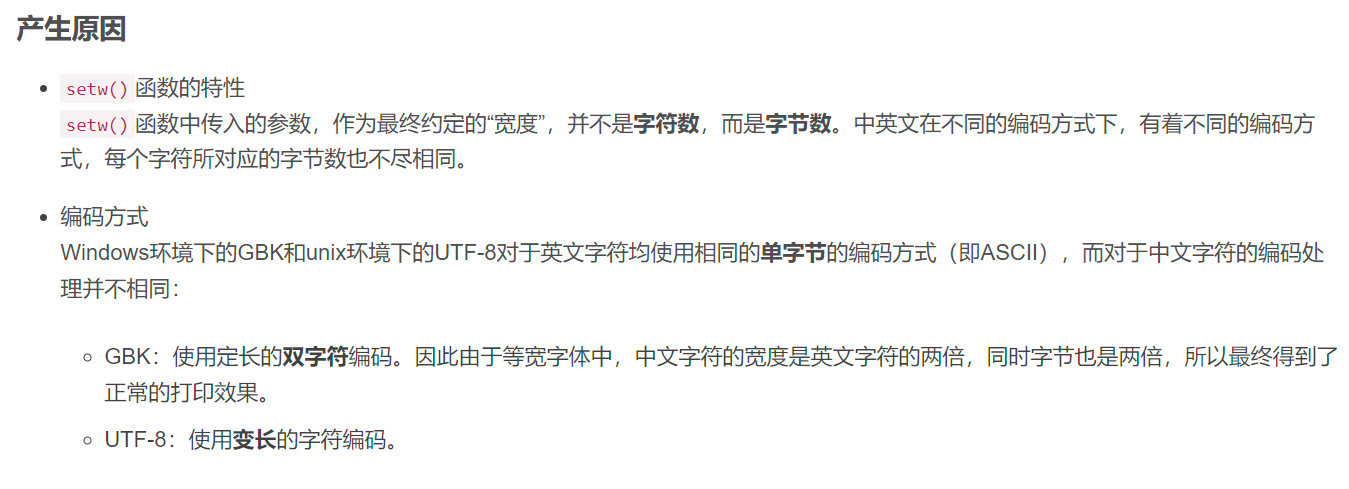
图 2查询模块流程示意图

**遇到的问题及解决方法如下：**

* + **问题1：**

**问题描述：**中英文混合输出时，输出来的文字始终无法对齐。

**解决方法：**在网上找到因为编码问题和setw函数特性导致程序输出全角字符无法像正常英文一样通过setw控制输出格式。

[](https://blog.csdn.net/Elko_265/article/details/123949243)

起初想过通过使用宽字符串存储或是将项目编码转为GBK，但由于项目大部分已经完成，重构难度较大，最后选择通过计算字符串中的全角字符数量动态调整输出格式。

* + **其他问题：**

**问题描述：**项目刚开始如何进行规划，如何设计用户友好的界面和提示词，开发过程中的版本控制……

1. **实验总结**

我设计的物业维修管理系统基本满足任务书的功能要求，类的结构和关系清晰，功能完善，而且在批量数据存储上采用了链表，在空间的利用率、集合数据操作（添加和删除）上具有优势。

这个项目做的并不是很完善，但基本上可以实现所要求的功能，小的缺陷还有很多，例如：没有对用户输入的数据进行合理性校验；没有测试奇葩数据对项目的影响；没有做出适合计算机新手的UI界面；遇到过长数据时输出格式可能会混乱；输出格式没有根据窗口大小动态调整；读入失败时会造成意想不到的BUG；没有使用数据库，对大量数据处理可能效率不高；在终端和程序中输出格式不同……

对于这个项目未来的展望：添加UI，不再使用命令行控制；对于所有要输入的数据进行合法检验；将归并排序的实现由递归改写为迭代，避免栈溢出。基本就是完善上面的小问题。

通过这次C++的大型实验，我深刻的明白到：课本知识与实践能力相结合的重要性。会读程序的人并不一定会编程序。要想把一门专业课程学好，必须增强自己的动手实践能力，而不是一天到晚只知道看书，那种行为只不过是“纸上谈兵”。看再多的书都不如自己亲手试一试。俗话说的好：会打仗的士兵才是好士兵。再者，课本上的知识不一定是完全准确的，只有自己动手进行试验过才知道。实践是检验真理的唯一标准，这话不假。

1. **附录：源代码**

[Main.cpp](main.cpp)

<Date.h>

<Date.cpp>

[List.cpp](list.cpp)

[List.h](list.h)

<RepairRecord.h>

<RepairRecord.cpp>