

简单个人图书管理系统

一、问题描述

同学们在学习过程中有很多书籍，对自己购买的书籍进行分类和统计是一种良好的学习习惯。如果采用文件来存储书籍号、书名、作者名、价格与购买日期等相关的书籍信息，辅之以程序对里面的书籍信息进行统计和查询，将使管理工作变得更轻松而有趣。

二、基本要求

1. 系统至少应具备如下的功能

- 存储书籍的各种相关信息
- 提供查找功能，按照书名或者作者名查找需要的书籍
- 提供插入、删除与更新功能
- 排序功能，按照作者名对所有的书籍进行排序，并按照排序后的结果进行显示。

2. 要求程序能按照书号、书名索引。

三、工具及准备工作

硬件：联想ThinkBook 16+

软件：VS 2022

四、分析与实现

需求分析：

1. 采用文件的形式存储书籍信息

- 记录保存在本地，在程序关闭后可以进行本地存储，下一次打开时能读取以前记录

2. 由于要求按书号、书名索引，书号为关键字，书名为次关键字，采用多重表文件方式组织索引

- 若书籍文件过大，无法一次性加载到内存，采用多重表文件可以有效进行检索咨询
3. 为接收文件中的内容，需要有结构来存储相应的内容，并且建立索引，还应建立相应的索引项结构。在查找和排序时需要对记录或关键字进行比较操作，为此重载相关的关系运算符。

下面对关键细节进行阐述：

图书管理类：

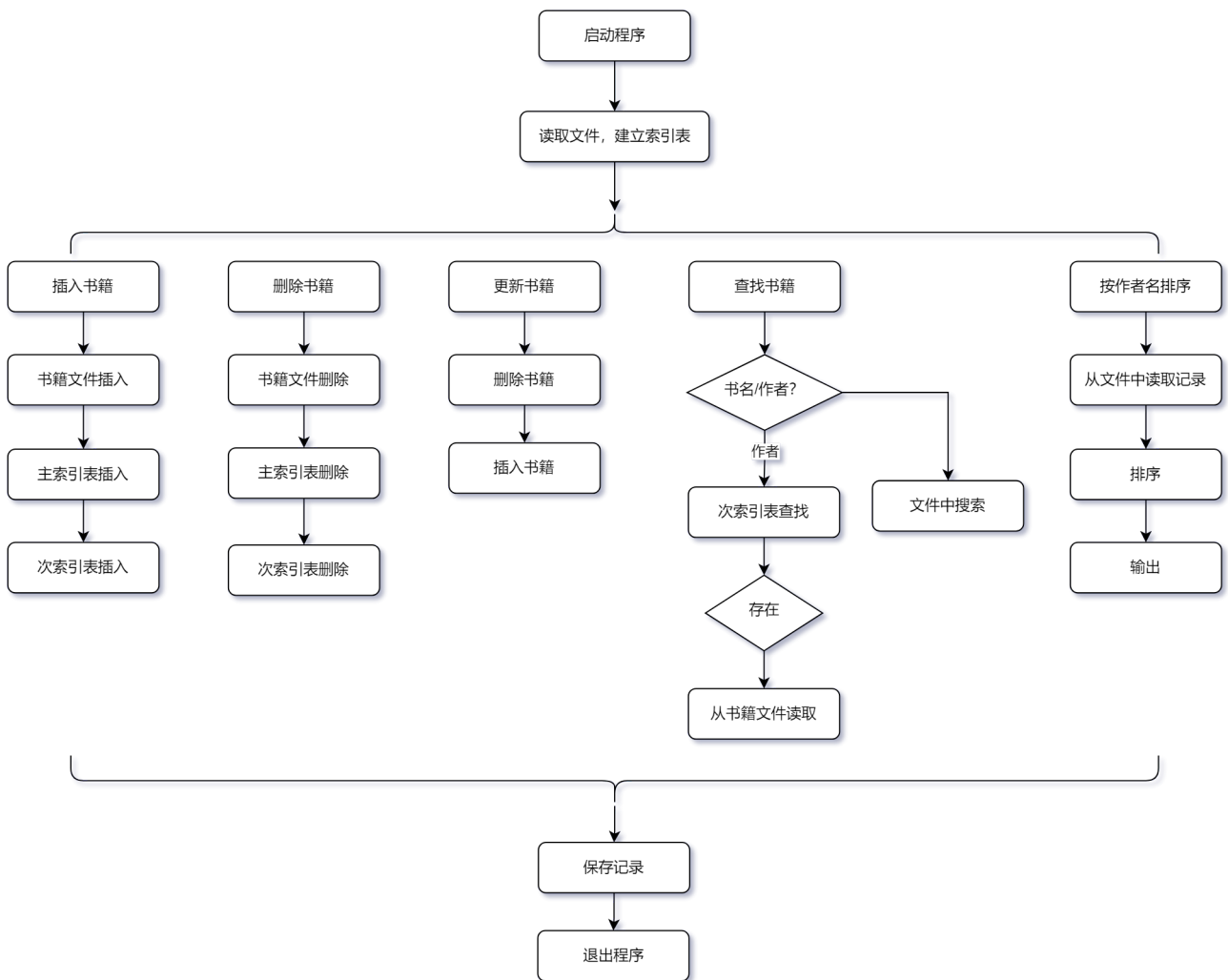
```
private:
    // 数据成员
    // 文件
    fstream BookFile; // 书籍文件
    // 索引表
    SqlList<IdIndex> IdIndexList; // ISBN号索引表，升序
    SqlList<NameIndex> NameIndexList; // 书名索引表，升序
```

BookFile 是书籍主文件，每次运行时，读取 **BookFile** 文件创建 **ISBN** 号主索引表和书名次索引表。

由于删除文件中间部分记录时需要对文件大量记录做相应调整，因此在存储数据的结构体中增加 **IsDelete** 变量用于记录是否被删除，删除时对文件中相应记录中 **IsDelete** 修改即可。

在程序运行结束时，更新书籍文件，将 **IsDelete** 为1的书籍记录剔除。

流程图如下：



五、测试与结论

1. 正常功能测试

对程序的基本功能进行测试，测试结果正常！

```
2856 4/5/2019: Visual Studio 调试
程序启动, 请稍等.....
索引表初始化完成
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 1
请输入书籍的ISBN号: 234
请输入书籍的书名: 计算机组成原理
请输入书籍的作者: 位置4
请输入书籍的价格: 150
请输入书籍的购买日期(year month day, 以空格隔开): 2019 7 20
插入书籍成功!
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 5
按作者名排序后的书籍信息如下:
ISBN号 书名 作者 价格 购买日期
123 数据结构 未知0 88.00 2020 9 30
789 算法设计 未知0 100.00 2022 4 4
456 数据结构 未知1 66.00 2018 10 1
1566 鱼不存在 未知2 50.00 2024 12 25
6541 离散数学 未知3 66.00 2023 8 1
234 计算机组成原理 位置4 150.00 2019 7 20
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 2
请输入要更新的书籍的ISBN号: 234
请输入书籍的ISBN号: 234
请输入书籍的书名: 计算机组成原理
请输入书籍的作者: 未知4
请输入书籍的价格: 150
请输入书籍的购买日期(year month day, 以空格隔开): 2019 7 20
更新书籍成功!
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 4
进入查找函数
1.按书名查找 2.按作者查找 3.退出查找
请输入选择: 1
请输入要查找的书名: 数据结构
查找到的书籍信息如下:
ISBN号 书名 作者 价格 购买日期
123 数据结构 未知0 88.00 2020 9 30
456 数据结构 未知1 66.00 2018 10 1
1.按书名查找 2.按作者查找 3.退出查找
请输入选择: 2
请输入要查找的作者: 未知0
书籍数量7
查找到的书籍信息如下:
ISBN号 书名 作者 价格 购买日期
123 数据结构 未知0 88.00 2020 9 30
789 算法设计 未知0 100.00 2022 4 4
1.按书名查找 2.按作者查找 3.退出查找
请输入选择: 3
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 2
请输入要删除的书籍的ISBN号: 234
删除书籍成功!
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 5
按作者名排序后的书籍信息如下:
ISBN号 书名 作者 价格 购买日期
123 数据结构 未知0 88.00 2020 9 30
789 算法设计 未知0 100.00 2022 4 4
456 数据结构 未知1 66.00 2018 10 1
1566 鱼不存在 未知2 50.00 2024 12 25
6541 离散数学 未知3 66.00 2023 8 1
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 6
成功退出系统!
E:\New Project(C++)\数据结构与算法\实验课\第五次实验\64\Debug\第五次实验.exe (进程 14636)已退出, 代码为 0。
按任意键关闭此窗口. . .
```

2. 异常输入测试

对各个分支的可能异常输入进行测试, 防止程序意外崩溃, 测试结构良好!

```
Microsoft Visual Studio 调试
程序启动, 请稍等.....
索引表初始化完成
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: sdags
输入错误, 请输入一个数字选项。
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 7
输入数字错误, 请重新输入
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 2
请输入要删除的书籍的ISBN号: asdgsd
删除书籍不存在!
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 3
请输入要更新的书籍的ISBN号: sagadf
更新书籍不存在!
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 4
进入查找函数
1.按书名查找 2.按作者查找 3.退出查找
请输入选择: sahqh
输入错误, 请输入一个数字选项。
1.按书名查找 2.按作者查找 3.退出查找
请输入选择: 4
输入数字错误, 请重新输入
1.按书名查找 2.按作者查找 3.退出查找
请输入选择: 3
请选择操作: 1.插入书籍 2.删除书籍 3.更新记录 4.查找书籍 5.按作者名排序 6.退出
请输入选择: 6
成功退出系统!
E:\New Project(C++)\数据结构与算法\实验课\第五次实验\64\Debug\第五次实验.exe (进程 13464)已退出, 代码为 0。
按任意键关闭此窗口. . .
```

六、思考与感悟

1. 输入的类型合法性检测:

在开发过程中, 用户输入的验证是非常重要的的一环, 它能够确保程序的健壮性和可靠性。对于 **Book** 类中的价格和年份等属性, 可以在输入时进行类型检查和范围验证。例如, 价格应该是一个非负数, 而年份应该是一个合理的年份值。可以通过正

则表达式来验证输入是否为数字，并且可以设置一个合理的范围来确保年份的合法性。此外，还可以提供错误提示，引导用户进行正确的输入。这样的错误处理机制不仅能够提高用户体验，还能避免程序因非法输入而出现异常。

2. 代码模块间耦合性：

代码的模块化是软件工程中的一个核心概念，它有助于提高代码的可维护性和可扩展性。例如，可以创建一个 `MultipleTableFile` 类，它作为基类，封装了与多重表文件相关的操作，如打开、关闭、读取和写入等。然后，可以创建具体的子类来处理不同类型的表文件，这样就能够将具体的实现细节与通用接口分离，降低耦合性。此外，继承和多态的使用可以让代码更加灵活，通过抽象基类定义通用接口，而具体的子类实现特定的行为，这样可以在不修改已有代码的情况下扩展新功能。

3. 对多重表文件数据结构的理解：

通过这次实验，了解到多重表文件是一种复杂的数据结构，它涉及到文件的组织和管理。这种结构通常用于数据库和文件系统中，以提高数据的存取效率。通过实验学会了如何设计和实现这种结构，包括如何维护索引、如何进行数据的插入和删除操作，以及如何优化数据的存储和检索。

4. 深化对git的使用与熟悉：

Git是一个强大的版本控制系统，它帮助开发者管理代码的变更历史，促进团队协作。掌握了如何使用git的基本命令，如 `commit`、`push`、`pull`、`branch` 和 `merge` 等。更好地跟踪代码的变更，以及如何与他人协作开发。学会了如何使用分支来并行开发新功能，以及如何合并这些分支而不会引起冲突。意识到了版本控制的重要性，以及它在软件开发中的作用。