



**程序设计课程设计报告**

**题 目 广工计算机学院C语言初学者学伴(智能)**

**学 院 计算机学院**

**专 业 计算机**

**年级班别 2024 级（ 12 ）班**

**学 号 3124004474**

**学生姓名 李忠林**

**指导教师 张静**

**成 绩**

|  |  |
| --- | --- |
| 程序功能完成情况 |  |
| 报告格式是否与要求相符 |  |
| 测试用例全面情况 |  |
| 报告内容是否准确全面 |  |

**2025 年 1月** **2日**

**题目**

**C 课程设计题目第一套 难度：1**

**题目：广工计算机学院C语言初学者学伴(智能)**

**一、 功能需求说明(必须采用结构体和动态链表实现)**

**1. 问题保存在文件question.txt中,格式如下：**

**每一条记录包括一个问题的问题编号、问题类型、问题内容、问题答案、首次查询日期、末次查询日期、查询次数、用户满意次数。**

**问题类型：**

**（1）anyview的常见问题，（2）智慧树的常见问题，（3）编译器的常见问题，（4）常见语法错误，（5）常见算法错误。**

**2．查询搜索功能**

**（1）用户能够通过输入问题类型、问题内容，查询到相应的答案**

**（2）用户可以自定义一个时间段进行查询问题的求助情况**

**（3）用户可以按查询次数查找求助比较多的问题**

**（4）管理员可查询任一用户的查询记录，可按照固定时段、自定义时段、问题类型、问题内容、已删除记录查询**

**3.排序功能：按查询次数、用户满意程度（=用户满意次数/查询次数）等中的一个进行（升序或降序）排序。**

**4．数据录入：管理员能录入收集到的问题，可以一次完成若干条记录的输入。**

**5.显示功能：可进行全部问题记录的显示，也可按照查询结果进行显示 。**

**6.数据变更：管理员可以更改或删除问题记录，但删除的信息要保存在另外一个文件 question\_undel.dat**

**7.统计功能：**

**（1）用户可统计自己在一个自定义时间段里的求助情况**

**（2）管理员可统计各个用户在一个自定义时间段里的求助情况**

**8. 用户和管理员的权限要分开。**

**二、加分项**

**1.使用图形界面。**

**2.使用线、图表示查询记录和统计消费结果**

**3.可参考实际情况对软件进行功能扩充（例如：能体现一定的智能功能，比如智能推荐、强化学习，具体请自行查阅资料）。**

**三、 设计要求**

**1.问题库文件为 question.txt，所有问题记录以问题编号从小到大排序（该文件自行建立）。要求编程序实现查询、排序、统计、录入、更新等功能。具体要求如下：**

**（1）要求显示如下界面**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1------查询 2------排序 3------统计 4------录入 5------更新**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**通过选择 1-5 来确定要做哪一个操作。**

**（2）若选择 1 则出现如下界面**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1-----按问题类型查询 2-----按问题内容查询 3-----按查询次数查询**

**4-----按固定时段查询 5-----按日期区间段查询**

**6-----已删除记录查询（管理员才有此项功能）**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**通过选择 1-6 来确定操作,有符合的记录输出到屏幕，若无，则输出相关提示信息。**

**2．若选 2，按查询次数、用户满意程度（=用户满意次数/查询次数）等中一个进行（升序或降序）排序。**

**3．若选 3/4/5/6，按照上面功能需求说明实现相应功能。**

**4. 不同的模块都要有出错处理，并能给出出错提示。如输入数据错误，文件操作错误等等。**

**5．以上各个功能均编写成子函数，有良好的注释说明，由主函数调用实现。**

**6．必须采用结构体和动态链表实现对记录数据的存储和访问。**

**7. 必须使用文件保存数据 。**

**8.应提供一个界面来调用各个功能,调用界面和各个功能的操作界面应尽可能清晰美观 。**

**二、测试数据要求**

**(1)不少于 3 个学生的查询记录；(2)不少于 10 个月的查询记录；**

**(3)每个月的查询不少于 5 个； (4)删除的记录不少于 10 条；**

**(5)每种问题类型（anyview/智慧树/编译器/常见语法错误/常见算法错误）至少有10个问题。**

**目录**

[**1、** **需求分析** 1](#_Toc187509350)

[**1.1系统及模块介绍** 1](#_Toc187509351)

[**1.2功能结构图** 3](#_Toc187509352)

[**2、总体设计方案** 4](#_Toc187509353)

[**2.1设计技术说明** 4](#_Toc187509354)

[**2.2结构体与变量设计说明** 7](#_Toc187509355)

[**2.3系统中函数的详细说明** 8](#_Toc187509356)

[**2.4函数关系示意图** 9](#_Toc187509357)

[**3、函数设计与实现** 10](#_Toc187509358)

[**3.1问题读取** 10](#_Toc187509360)

[**3.2问题查询** 11](#_Toc187509361)

[**3.3问题排序** 13](#_Toc187509362)

[**3.4查询次数统计** 14](#_Toc187509363)

[**3.5问题录入（管理员）** 15](#_Toc187509364)

[**3.6问题变更（管理员）** 16](#_Toc187509365)

[**4、系统测试** 17](#_Toc187509366)

[**5、系统使用说明** 21](#_Toc187509367)

[**6、总结** 22](#_Toc187509368)

[**参考文献** 24](#_Toc187509369)

[**附录（代码）** 25](#_Toc187509370)

**1 需求分析**

**1.1系统及模块介绍**

初学者学伴一款以信息管理为核心的计算机应用系统，旨在收集、处理常见问题相关信息，为用户提供便捷的问题查询服务，同时为管理员提供全面的数据管理功能。相较于传统管理方式，该系统通过有效的程序结构支持各类数据操作，显著提升管理效率，实现常见问题管理的系统化、规范化与自动化。系统主要分为用户部分和管理员部分：

1.1.1用户部分

（1）查询模块

功能描述：用户可通过此模块依据多种条件查询问题。模块提供按问题类型（如anyview常见问题、智慧树常见问题等）、问题内容（关键词）、查询次数范围、固定时段以及日期区间段等方式进行查询。查询到问题后，会显示问题的详细信息，并记录查询次数及用户是否满意本次查询。若用户满意，会增加问题的用户满意次数。

应用场景：当用户遇到问题时，可根据自身需求，选择合适的查询条件快速找到相关问题及答案。例如，用户在使用某软件过程中遇到特定类型问题，可通过问题类型查询获取解决方案；若对问题描述有模糊印象，可通过问题内容关键词查询。

（2）排序模块

功能描述：用户能够选择依据查询次数或用户满意程度对问题进行排序查看。系统会根据用户选择的排序依据，对问题链表进行相应排序，并展示排序后的所有问题。

应用场景：用户希望了解哪些问题被查询得最多，或者哪些问题的用户满意度最高时，可通过此模块进行排序查看，以便优先关注热门或优质问题。

（3）统计模块（用户端）

功能描述：用户可在此模块统计自己在自定义时间段内的求助情况，即统计该时间段内问题的总查询次数。

应用场景：用户想要了解自己在一段时间内对系统的使用频率，或评估自己在特定时期内遇到问题的数量时，可使用此功能进行统计。

1.1.2管理员部分

（1）登陆及权限判断模块

功能描述：管理员通过`isAdm`函数进行身份验证。选择管理员登录后，需输入密码（初始预设为`123abc`），若密码正确则获得管理员权限，可执行后续管理操作；若密码错误，可选择重新登录。多次错误后则默认以普通用户身份使用系统。

应用场景：确保系统管理的安全性，只有授权的管理员能够执行敏感操作，防止未经授权的人员篡改数据或获取敏感信息。

（2）数据录入模块

功能描述：管理员具备录入新问题的权限。在该模块中，管理员可输入要录入的问题记录条数，然后依次填写每个问题的编号、类型、内容、答案等详细信息。系统会自动获取当前时间作为首次和末次查询日期，并将新问题数据写入文件和添加到问题链表中。

应用场景：当系统中出现新的常见问题时，管理员可及时将其录入系统，以便用户能够查询到相关答案。

（3）菜单编辑模块（类比问题数据变更模块）

功能描述：此模块具备更改和删除问题数据的功能。管理员可选择更改问题记录，输入要更改的问题编号，找到对应问题后修改其内容和答案，并同步更新文件。也可选择删除问题记录，输入要删除的问题编号，系统会删除对应的链表节点，并将删除的问题信息保存到另一个文件（`question\_undel.dat`），同时更新原问题文件。

应用场景：当问题的答案需要更新，或者某些问题不再适用时，管理员可通过此模块进行相应的修改或删除操作，保证系统中问题数据的准确性和时效性。

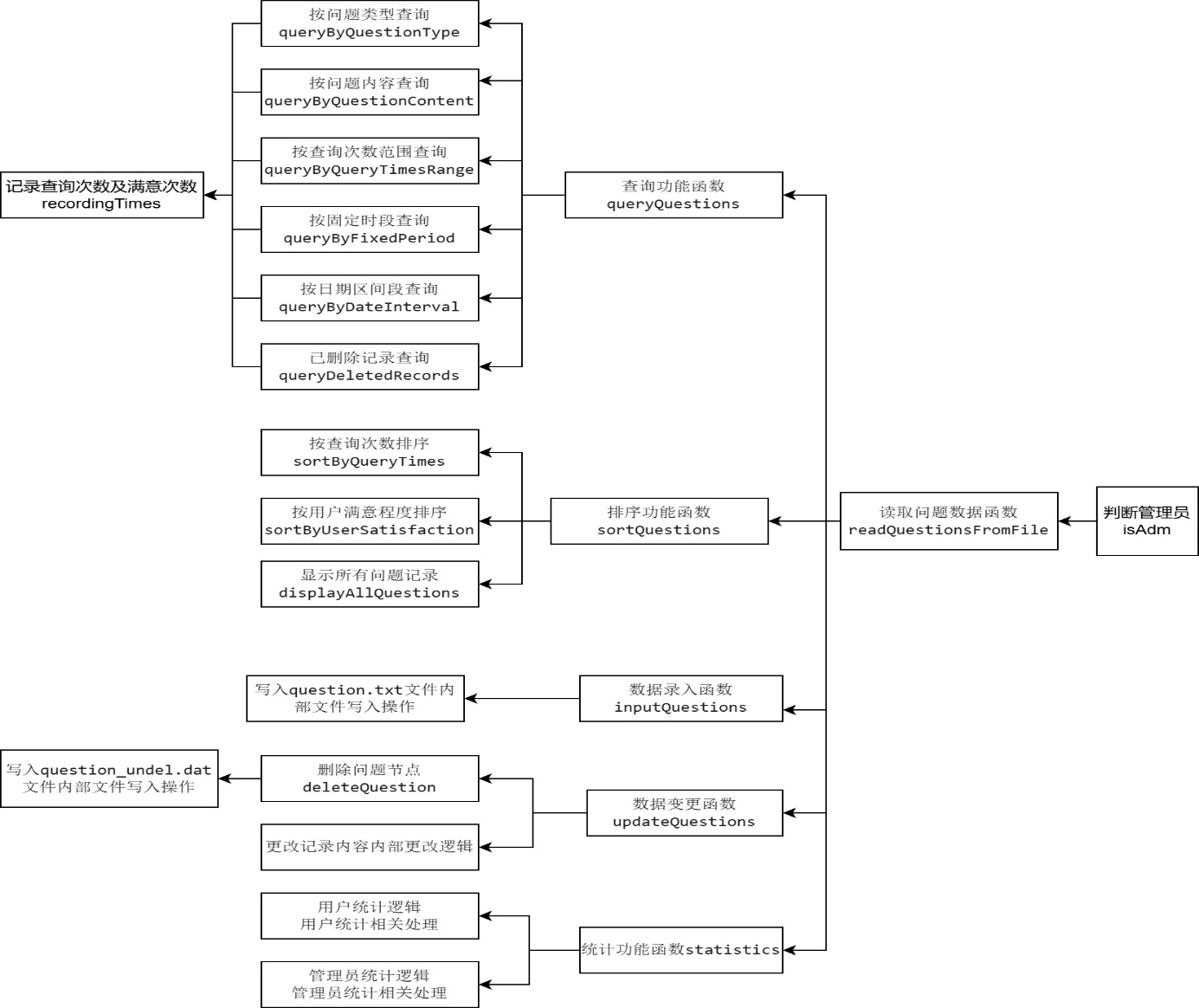
（5）菜单查询模块（类比问题查询模块）

功能描述：管理员可通过此模块按多种条件查询问题，查询方式与用户端类似，但管理员还拥有查询已删除记录的权限（用户端无此权限）。查询结果会显示问题的详细信息，并记录查询相关数据。

应用场景：管理员在处理用户反馈、优化问题库等工作中，需要全面查询问题信息，包括已删除的问题记录，以便进行数据分析和问题管理。

总体而言，常见问题管理系统通过清晰的用户和管理员功能划分，为用户提供了便捷的问题查询服务，为管理员提供了高效的数据管理手段，有助于提升常见问题管理的整体效率和质量。

**1.2功能结构图**

****

**2 总体设计方案**

**2.1设计技术说明**

2.1.1系统概述

常见问题管理系统旨在为用户提供便捷的问题查询服务，同时为管理员提供全面的数据管理功能，以实现常见问题管理的系统化、规范化与自动化。系统主要分为用户部分和管理员部分，通过有效的程序结构支持各类数据操作。

2.1.2技术架构

（1）编程语言：采用 C 语言进行开发，C 语言具有高效、灵活且贴近底层的特点，能够满足系统对性能和资源控制的需求。

（2）数据存储

链表：使用链表结构存储问题数据。链表的动态特性使得在运行时能够方便地添加、删除和查找节点，适合管理不断变化的问题数据集合。每个节点对应一个问题，包含问题的各种属性，如问题编号、类型、内容、答案、查询日期、查询次数和用户满意次数等。

文件：利用文件进行数据的持久化存储。程序启动时从文件读取数据构建链表，运行过程中对数据的修改会同步回写到文件。主要涉及两个文件：`question.txt`用于存储当前有效的问题数据，`question\_undel.dat`用于保存已删除的问题记录（仅管理员相关操作涉及）。

2.1.3模块设计与实现

用户部分

（1）查询模块

功能：支持按多种条件查询问题，包括问题类型、问题内容、查询次数范围、固定时段和日期区间段。

实现：

按问题类型查询：`queryByQuestionType`函数遍历链表，比较每个节点的`questionType`属性，若匹配则显示问题详细信息，并调用`recordingTimes`函数记录查询次数和用户满意度。

按问题内容查询：`queryByQuestionContent`函数通过`strstr`函数在问题内容中查找用户输入的关键词，找到匹配项后进行相应处理。

按查询次数范围查询：`queryByQueryTimesRange`函数在链表遍历过程中，判断节点的`queryTimes`属性是否在指定范围内，满足条件则显示并记录相关信息。

按固定时段和日期区间段查询：`queryByFixedPeriod`和`queryByDateInterval`函数类似，通过比较节点的`firstQueryDate`和`lastQueryDate`与用户输入的日期范围，确定是否显示该问题。

（2）排序模块

功能：允许用户按查询次数或用户满意程度对问题进行排序。

实现：

按查询次数排序：`sortByQueryTimes`函数采用冒泡排序算法，比较相邻节点的`queryTimes`属性，若顺序不符合要求则交换节点数据。若查询次数相同，则进一步比较`userSatisfiedTimes`属性。

按用户满意程度排序：`sortByUserSatisfaction`函数同样使用冒泡排序，计算每个节点的用户满意程度（`userSatisfiedTimes`与`queryTimes`的比值），并以此为依据进行排序。若满意程度相同，则比较`queryTimes`属性。

（3）统计模块（用户端）

功能：统计用户在自定义时间段内的问题总查询次数。

实现：`statistics`函数在用户端的逻辑中，遍历链表，比较每个节点的`firstQueryDate`和`lastQueryDate`与用户输入的日期范围，若在范围内则累加`queryTimes`属性值。

管理员部分

（1）登陆及权限判断模块

功能：判断用户是否为管理员，通过输入密码进行身份验证。

实现：`isAdm`函数提示用户是否进行管理员登录，若选择是则要求输入密码。与预设密码（`123abc`）比对，正确则设置`isAdmin`标志为 1，赋予管理员权限；错误则可选择重新登录，多次错误后默认以普通用户身份运行。

（2）数据录入模块

功能：管理员可录入新的问题数据，并将其添加到链表和文件中。

实现：`inputQuestions`函数首先检查`isAdmin`标志，确认管理员权限后，提示输入要录入的记录条数。然后循环读取每个问题的各项信息，使用`time`函数获取当前时间作为首次和末次查询日期，初始化查询次数和用户满意次数为 0。将新问题数据写入文件，并创建新的链表节点插入到链表头部。

（3）数据变更模块（类比订餐系统的菜单编辑模块）

功能：支持管理员更改问题的内容和答案，或删除指定问题，并更新文件。

实现：

更改记录：`updateQuestions`函数在管理员选择更改记录时，提示输入要更改的问题编号。找到对应节点后，提示输入新的问题内容和答案，然后更新文件。

删除记录：`deleteQuestion`函数用于删除指定问题编号的节点。先查找节点，找到后将其从链表中移除，同时将节点信息写入`question\_undel.dat`文件，然后更新`question.txt`文件以去除已删除记录。

（4）查询模块（管理员扩展功能）

功能：除了与用户端相同的查询功能外，管理员还可查询已删除的问题记录。

实现：`queryDeletedRecords`函数在管理员选择查询已删除记录时，打开`question\_undel.dat`文件，读取并显示其中保存的已删除问题记录。

2.1.4文件操作

（1）读取文件：`readQuestionsFromFile`函数使用`fopen`以只读模式打开`question.txt`文件，通过`fscanf`逐行读取文件内容，为每个问题创建链表节点并添加到链表中，最后关闭文件。

（2）写入文件：在数据录入、更新和记录查询信息等操作中，使用`fopen`以写入或追加模式打开文件，通过`fprintf`将数据写入文件，操作完成后关闭文件。例如，`inputQuestions`函数以追加模式打开`question.txt`文件写入新问题；`updateQuestions`函数在更改或删除问题后，重新以写入模式打开`question.txt`文件更新所有问题数据。

2.1.5输入验证与错误处理

（1）输入验证：在关键输入处进行了一定的验证。如在`inputQuestions`函数中，输入问题类型时，检查输入值是否在有效范围内（1 - 5），若不在则提示重新输入。

（2）错误处理：文件操作失败时进行相应提示。例如，`readQuestionsFromFile`函数中若无法打开文件，会提示用户检查文件是否存在及权限设置；`inputQuestions`和`updateQuestions`等函数中文件写入失败时，会使用`perror`输出错误信息。但在某些复杂情况下，如内存分配失败时，虽然有提示但未进行全面的资源清理，可能存在潜在的内存泄漏风险。

**2.2结构体与变量设计说明**

typedef struct Question {

int questionId; // 问题编号

int questionType; // 问题类型

char questionContent[100]; // 问题内容

char questionAnswer[301]; // 问题答案

char firstQueryDate[20]; // 首次查询日期

char lastQueryDate[20]; // 末次查询日期

int queryTimes; // 查询次数

int userSatisfiedTimes; // 用户满意次数

struct Question \*next; // 指向下一个节点的指针，用于链表

} Question;

**2.3系统中函数的详细说明**

Question \*readQuestionsFromFile(const char \*filename);

// 从文件读取问题数据到链表

void queryQuestions(Question \*head);

// 根据选择执行相应查询功能的函数

void sortQuestions(Question \*head);

// 排序功能函数，按指定规则排序链表

void inputQuestions(Question \*\*head);

// 数据录入函数，管理员录入新问题

void displayAllQuestions(Question \*head);

// 显示所有问题记录函数

void updateQuestions(Question \*\*head);

// 数据变更（更改或删除）函数，管理员操作

void statistics(Question \*head);

// 统计功能函数，区分用户和管理员统计

void isAdm();

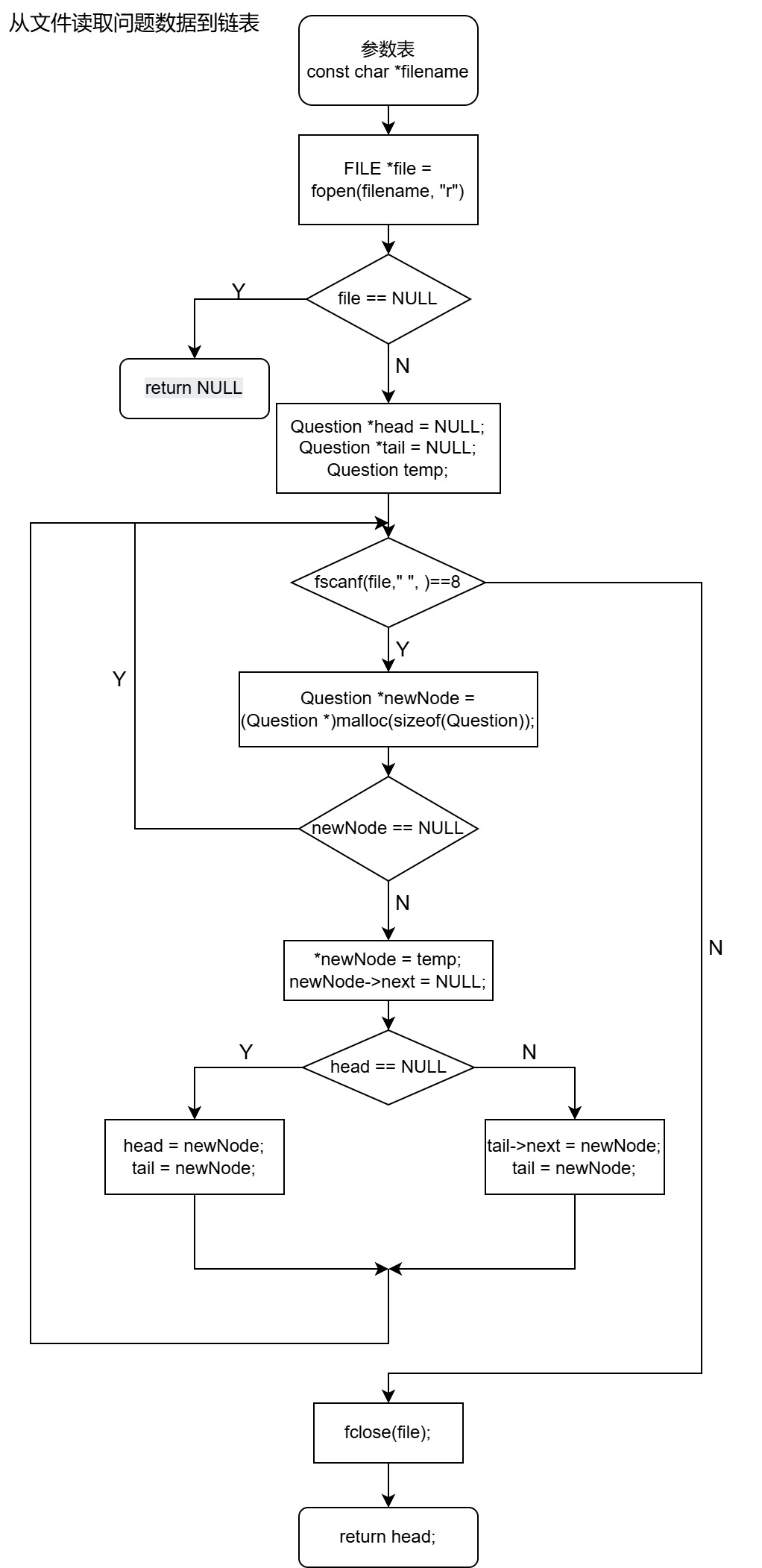
// 判断是否为管理员权限的函数

**2.4函数关系示意图**

**图示

描述已自动生成**

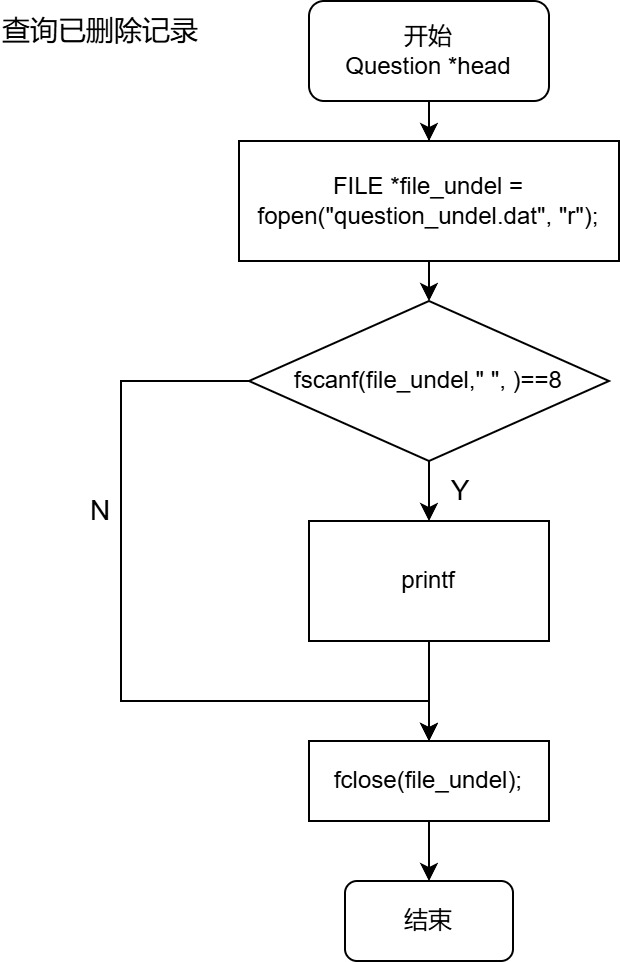
**3 函数设计与实现**

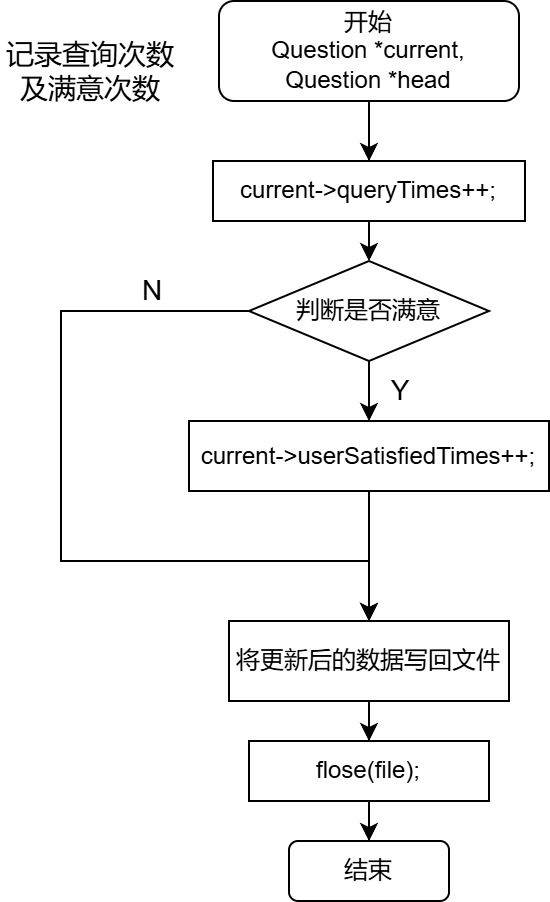
**3.1从文件读取问题数据**

**3.2问题查询**

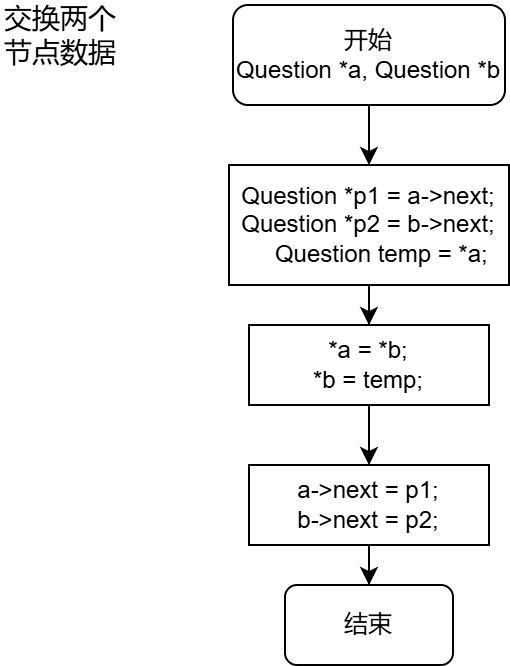
图示

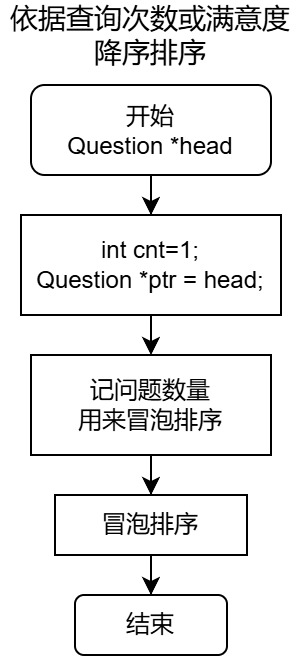
描述已自动生成



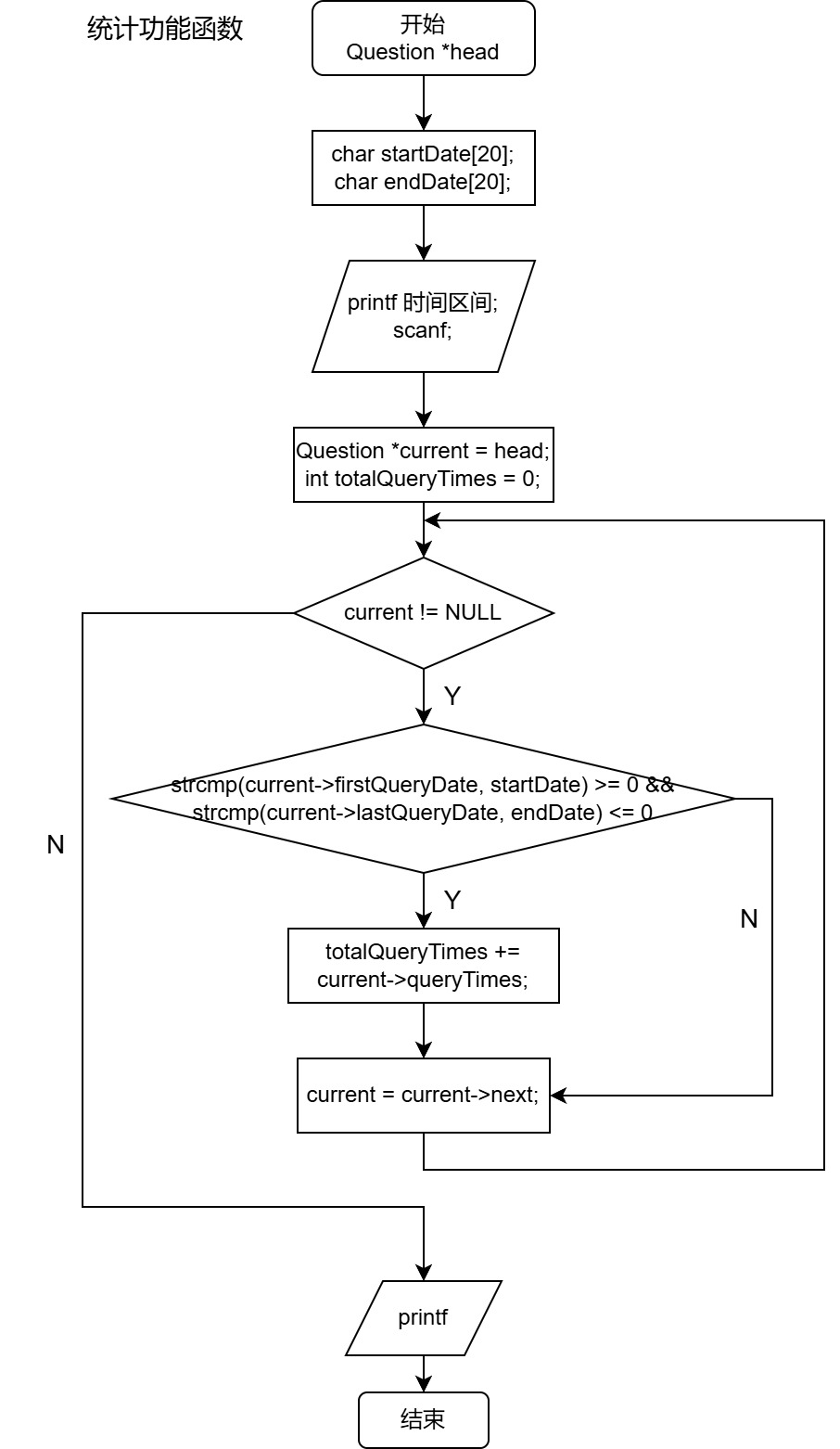


**3.3问题排序**

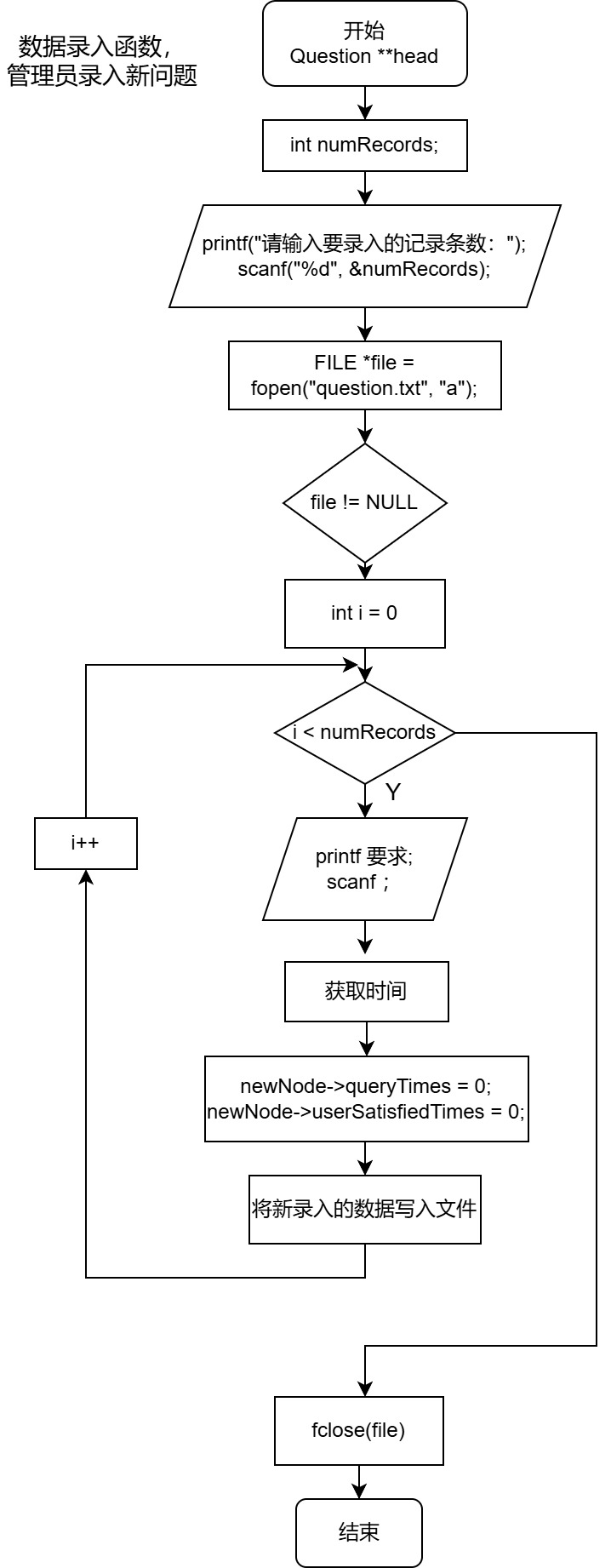


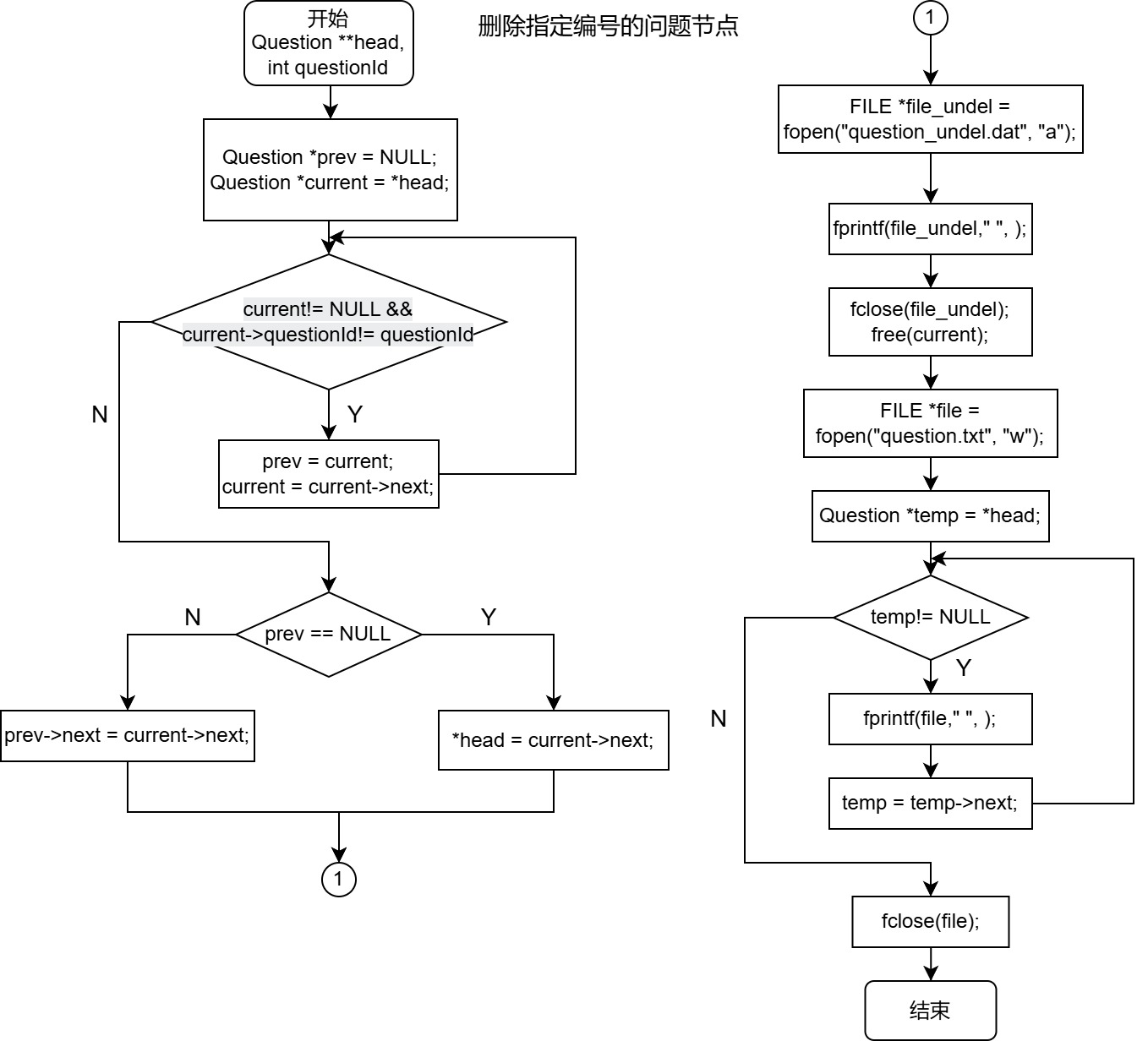


**3.4查询次数统计**



**3.5问题录入（管理员）**



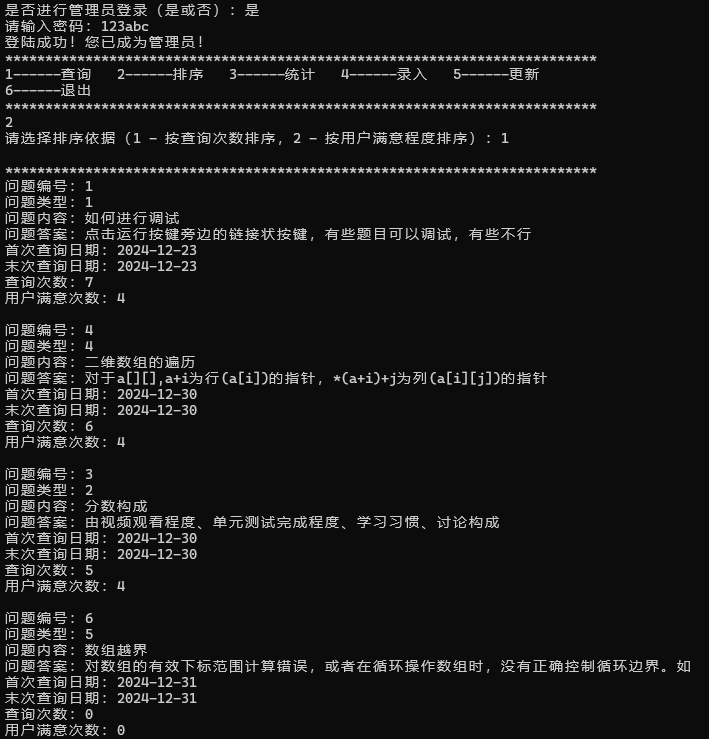
**3.6问题变更（管理员）**

**4 系统测试**

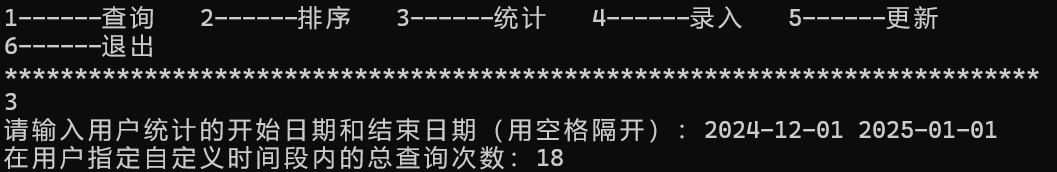
**4.1测试查询功能**



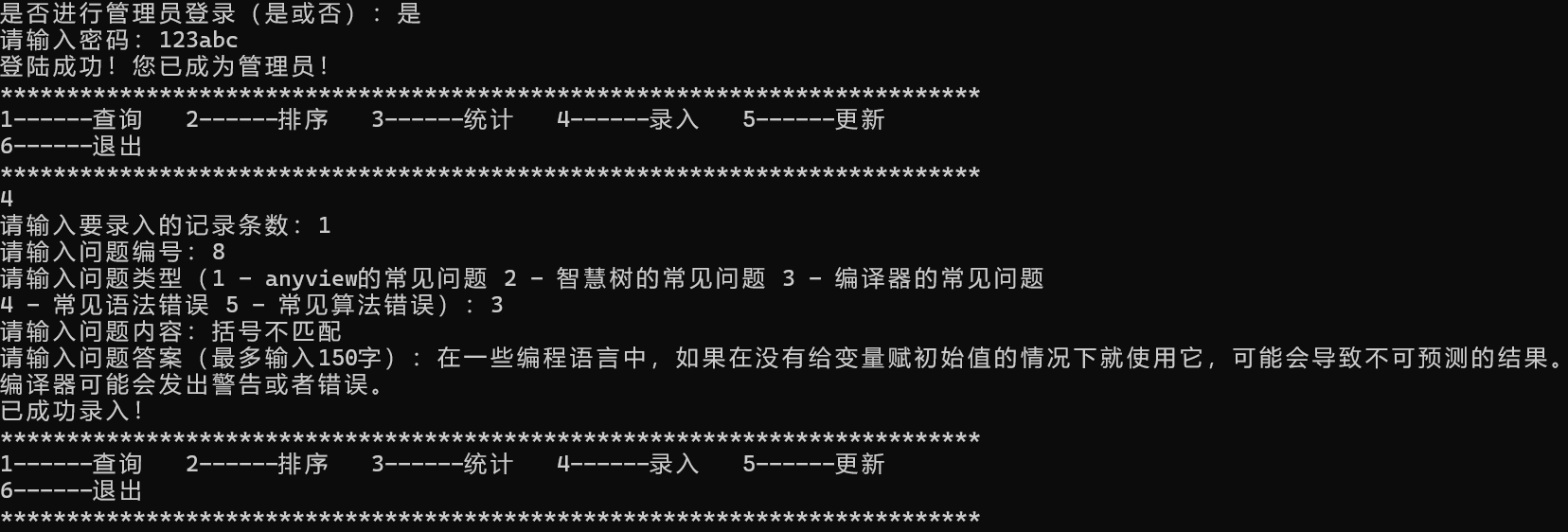
**4.2测试排序功能**



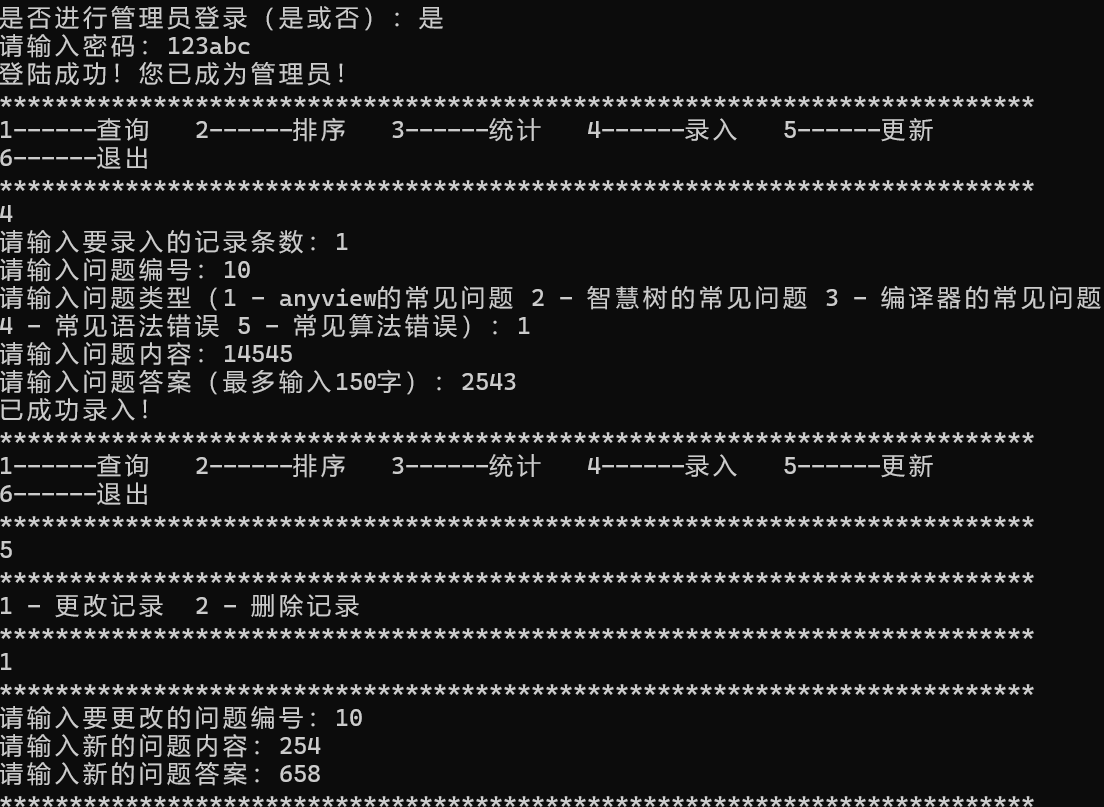
**4.3测试统计功能**

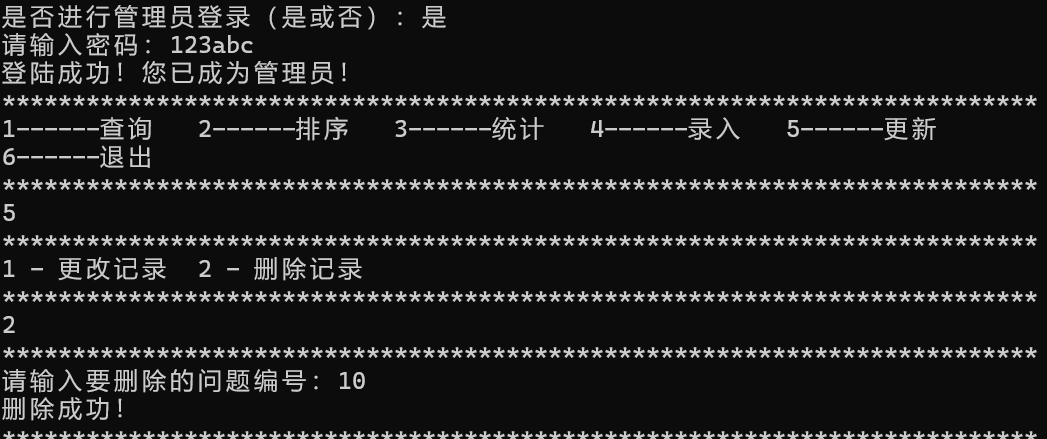


**4.4测试录入功能**



**4.5测试更新功能**





**5 系统使用说明**

1、确认是否登录管理员账户。（密码：123abc）

2、本应用有5大功能：查询、排序、统计、录入、更新。其中录入、更新和查询中的查询删除记录的功能只有管理员可以操作。

3、查询： （1）按问题类型查询：1 - anyview的常见问题 2 - 智慧树的常见问题 3 - 编译器的常见问题 4 - 常见语法错误 5 - 常见算法错误。

（2）按问题内容查询：可输入关键字进行查询。

（3）按查询次数查询：输入查询次数的最小值和最大值进行查询。

（4）按固定时段查询：输入固定时段的开始日期和结束日期。

（5）按日期区间段查询：输入日期区间段的开始日期和结束日期。

（6）已删除记录查询（管理员才有此项功能）：可查询删除记录。

4、排序：可按查询次数或满意度进行降次排序并查看。

5、统计：管理员可查看某段时间内所有问题的访问次数之和；用户和查看某段时间内自己查询的次数。

6、录入：管理员能录入收集到的问题，可以一次完成若干条记录的输入。

7、更新：可进行更改问题、删除问题的操作。

**6 总结**

**一、课程设计完成的功能**

1. **数据读取与存储：**程序能够从名为 question.txt 的文件中读取问题数据，并将其存储在链表结构中。每个问题包含问题编号、类型、内容、答案、首次查询日期、末次查询日期、查询次数以及用户满意次数等信息。在程序运行过程中对数据的修改，如录入新问题、更新问题、删除问题等操作，也会同步更新到文件中，确保数据的持久化。
2. **查询功能：**提供了多种查询方式，包括按问题类型查询、按问题内容模糊查询、按查询次数范围查询、按固定时段查询、按日期区间段查询。对于管理员用户，还额外提供了查询已删除记录的功能。在每次查询到相关问题后，会记录查询次数，并询问用户是否满意，若满意则增加用户满意次数，同时将更新后的数据写回文件。
3. **排序功能：**支持按查询次数降序排序和按用户满意程度降序排序。用户可以根据需求选择相应的排序依据，排序后会显示所有问题记录，方便用户查看不同排序规则下的问题列表。
4. **数据录入：**管理员用户具备录入新问题的权限。在录入时，需要输入问题的各项信息，包括问题编号、类型、内容、答案等，系统会自动获取当前时间作为首次查询日期和末次查询日期，并将新问题追加到文件 question.txt 中，同时更新链表结构。
5. **数据变更：**管理员可以对问题记录进行更改或删除操作。更改记录时，通过输入问题编号找到对应的问题，然后修改其内容和答案，并更新文件。删除记录时，不仅会从链表中移除该问题节点，还会将删除的信息保存到 question\_undel.dat 文件中，同时更新原文件 question.txt 以去除已删除的记录。
6. **统计功能：**区分管理员和普通用户的统计功能。管理员可以统计自定义时间段内的总查询次数，用于整体了解系统的使用情况。普通用户则可以统计自己在自定义时间段内的求助次数，方便用户了解自身的查询行为。

**三、需要改进的地方**

1. **内存管理优化**：在频繁进行数据操作（如多次录入、删除、排序等）时，内存的分配和释放操作较多，容易出现内存泄漏等问题。可以进一步优化内存管理机制，例如采用内存池等技术来提高内存使用效率，减少潜在的内存错误隐患。
2. **代码结构优化**：随着功能的增加，部分函数的代码逻辑变得较为复杂，可读性和可维护性有所下降。可以考虑对代码进行模块化重构，将一些关联性强的操作进一步封装成独立的模块，清晰划分各模块的职责，便于后续的功能扩展和代码维护。

**四、通过课程设计的收获**

1. **知识巩固与实践结合**：将之前所学的 C 语言基础知识，如结构体、链表操作、文件读写、函数调用等，进行了综合运用，通过实际的项目开发，加深了对这些知识的理解，也明白了如何将理论知识转化为实际可运行的程序。
2. **程序设计能力提升**：从系统功能的规划、模块的划分，到具体函数的实现以及各功能之间的交互逻辑设计，锻炼了整体的程序设计思维，学会如何构建一个相对完整、功能丰富的小型系统，提高了自己解决复杂问题的能力。
3. **调试技能培养**：在编写和运行程序过程中不可避免地遇到了各种错误和问题，通过不断地调试（如使用打印语句、单步调试等方式）去查找问题根源，定位并修复错误，掌握了更多调试技巧，也提升了面对问题时的耐心和细心程度。

**五、调试程序的体会**

1. **定位问题的复杂性**：有时候程序出现的错误现象并不能直接反映出问题的根源所在，比如一个看似是查询结果不正确的问题，可能是数据读取、链表遍历、条件判断等多个环节中的某一处出现了错误，需要通过逐步排查、添加调试输出语句等方式，从不同角度去观察程序的运行状态，才能准确找到问题所在。
2. **调试工具的重要性**：熟练掌握调试工具（如编译器自带的调试功能）能大大提高调试效率，例如可以方便地查看变量的值、跟踪函数的调用过程等，通过单步调试能清晰地看到程序的执行流程，对比预期的执行顺序和实际情况，快速定位逻辑错误点。
3. **耐心与细心是关键**：调试过程往往是一个反复尝试、不断分析的过程，需要保持足够的耐心，不能因为一时找不到问题而急躁。同时，每一个细节都可能影响结果，比如变量类型的匹配、语句的顺序等，都需要细心地去检查，任何一个小疏忽都可能导致程序出现异常。

总之，通过这次课程设计，不仅实现了一系列实用的功能，更在多方面得到了锻炼和提升，同时也认识到了程序设计中存在的不足以及需要继续改进的方向，为今后进一步的学习和开发积累了宝贵经验。

**参考文献**

[1] 谭浩强.C程序设计（第四版）[M]．北京:清华大学出版社.2010

[2] 谭浩强.C程序设计（第四版）学习辅导[M]．北京:清华大学出版社.2010

[3] 王芳.C语言程序设计[M]．北京：高等教育出版社.2011

**附录（代码）**

**以下为本程序的所有代码**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h> // 用于获取当前时间，在示例中用于首次/末次查询日期的记录

int isAdmin=0; // 默认不是管理员

// 定义问题结构体

typedef struct Question {

int questionId; // 问题编号

int questionType; // 问题类型

char questionContent[100]; // 问题内容

char questionAnswer[301]; // 问题答案

char firstQueryDate[20]; // 首次查询日期

char lastQueryDate[20]; // 末次查询日期

int queryTimes; // 查询次数

int userSatisfiedTimes; // 用户满意次数

struct Question \*next; // 指向下一个节点的指针，用于链表

} Question;

Question \*readQuestionsFromFile(const char \*filename); // 从文件读取问题数据到链表

void queryQuestions(Question \*head); // 根据选择执行相应查询功能的函数

void sortQuestions(Question \*head); // 排序功能函数，按指定规则排序链表

void inputQuestions(Question \*\*head); // 数据录入函数，管理员录入新问题

void displayAllQuestions(Question \*head); // 显示所有问题记录函数

void updateQuestions(Question \*\*head); // 数据变更（更改或删除）函数，管理员操作

void statistics(Question \*head); // 统计功能函数，区分用户和管理员统计

void isAdm();// 判断是否为管理员权限的函数

// 主函数

int main() {

// 判断是否为管理员

isAdm();

//isAdmin=1;

// 从文件读取问题数据到链表

Question \*questionList = readQuestionsFromFile("C:\\Users\\AijiA\\Desktop\\24级12班 3124004474 李忠林\\2、源程序及所需文件\\question.txt");

int choice;

do {

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("1------查询 2------排序 3------统计 4------录入 5------更新 \n6------退出\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

scanf("%d", &choice);

switch (choice) {

case 1:

// 查询

queryQuestions(questionList);

break;

case 2:

// 排序

sortQuestions(questionList);

break;

case 3:

// 统计功能函数，区分用户和管理员统计

statistics(questionList);

break;

case 4:

// 数据录入函数，管理员录入新问题

inputQuestions(&questionList);

break;

case 5:

// 数据变更（更改或删除）函数，管理员操作

updateQuestions(&questionList);

break;

case 6:

// 退出

break;

default:

printf("无效的选择，请重新输入！\n");

}

} while (choice >= 1 && choice <= 5);

// 释放链表内存

Question \*current = questionList;

Question \*next;

while (current!= NULL) {

next = current->next;

free(current);

current = next;

}

return 0;

}

// 从文件读取问题数据到链表

Question \*readQuestionsFromFile(const char \*filename) {

FILE \*file = fopen(filename, "r");

if (file == NULL) {

printf("无法打开文件 %s，请检查文件是否存在及权限设置！\n", filename);

return NULL;

}

Question \*head = NULL;

Question \*tail = NULL;

Question temp;

while (fscanf(file, "%d %d %s %s %s %s %d %d", &temp.questionId, &temp.questionType,

temp.questionContent, temp.questionAnswer, temp.firstQueryDate,

temp.lastQueryDate, &temp.queryTimes, &temp.userSatisfiedTimes) == 8) {

Question \*newNode = (Question \*)malloc(sizeof(Question));

if (newNode == NULL) {

printf("内存分配失败！\n");

continue;

}

\*newNode = temp;

newNode->next = NULL;

if (head == NULL) {

head = newNode;

tail = newNode;

} else {

tail->next = newNode;

tail = newNode;

}

}

fclose(file);

return head;

}

// 记录查询次数及满意次数

void recordingTimes(Question \*current, Question \*head) {

current->queryTimes++;

char satisDegree[]="是";

char isSatis[3];

printf("是否满意本次查询（是或否）：");

scanf("%s",isSatis);

if(strcmp(satisDegree, isSatis) == 0){

current->userSatisfiedTimes++;

}

// 将更新后的数据写回文件

FILE \*file = fopen("C:\\Users\\AijiA\\Desktop\\24级12班 3124004474 李忠林\\2、源程序及所需文件\\question.txt", "w");

if (file == NULL) {

perror("无法打开文件 question.txt 进行写入");

return;

}

Question \*temp = head;

while (temp!= NULL) {

if (fprintf(file, "%d %d %s %s %s %s %d %d\n", temp->questionId, temp->questionType,

temp->questionContent, temp->questionAnswer, temp->firstQueryDate,

temp->lastQueryDate, temp->queryTimes, temp->userSatisfiedTimes) < 0) {

perror("写入 question.txt 文件失败");

break;

}

temp = temp->next;

}

fclose(file);

}

// 按问题类型查询函数（辅助函数，供查询功能调用）

void queryByQuestionType(Question \*head, int questionType) {

Question \*current = head;

int found = 0;

while (current!= NULL) {

if (current->questionType == questionType) {

printf("问题编号：%d\n问题类型：%d\n问题内容：%s\n问题答案：%s\n首次查询日期：%s\n末次查询日期：%s\n查询次数：%d\n用户满意次数：%d\n",

current->questionId, current->questionType, current->questionContent,

current->questionAnswer, current->firstQueryDate, current->lastQueryDate,

current->queryTimes, current->userSatisfiedTimes);

found = 1;

recordingTimes(current,head);

}

current = current->next;

}

if (!found) {

printf("未找到该问题类型的记录！\n");

}

}

// 按问题内容模糊查询函数（辅助函数，供查询功能调用）

void queryByQuestionContent(Question \*head, const char \*content) {

Question \*current = head;

int found = 0;

while (current!= NULL) {

if (strstr(current->questionContent, content)!= NULL) {

printf("问题编号：%d\n问题类型：%d\n问题内容：%s\n问题答案：%s\n首次查询日期：%s\n末次查询日期：%s\n查询次数：%d\n用户满意次数：%d\n",

current->questionId, current->questionType, current->questionContent,

current->questionAnswer, current->firstQueryDate, current->lastQueryDate,

current->queryTimes, current->userSatisfiedTimes);

found = 1;

recordingTimes(current,head);

}

current = current->next;

}

if (!found) {

printf("未找到包含该内容的问题记录！\n");

}

}

// 按查询次数范围查询函数（辅助函数，供查询功能调用）

void queryByQueryTimesRange(Question \*head, int minTimes, int maxTimes) {

Question \*current = head;

int found = 0;

while (current!= NULL) {

if (current->queryTimes >= minTimes && current->queryTimes <= maxTimes) {

printf("问题编号：%d\n问题类型：%d\n问题内容：%s\n问题答案：%s\n首次查询日期：%s\n末次查询日期：%s\n查询次数：%d\n用户满意次数：%d\n",

current->questionId, current->questionType, current->questionContent,

current->questionAnswer, current->firstQueryDate, current->lastQueryDate,

current->queryTimes, current->userSatisfiedTimes);

found = 1;

recordingTimes(current,head);

}

current = current->next;

}

if (!found) {

printf("未找到查询次数在指定范围内的问题记录！\n");

}

}

// 按固定时段查询函数（辅助函数，供查询功能调用）

void queryByFixedPeriod(Question \*head, const char \*startDate, const char \*endDate) {

Question \*current = head;

int found = 0;

while (current!= NULL) {

if (strcmp(current->firstQueryDate, startDate) >= 0 && strcmp(current->lastQueryDate, endDate) <= 0) {

printf("问题编号：%d\n问题类型：%d\n问题内容：%s\n问题答案：%s\n首次查询日期：%s\n末次查询日期：%s\n查询次数：%d\n用户满意次数：%d\n",

current->questionId, current->questionType, current->questionContent,

current->questionAnswer, current->firstQueryDate, current->lastQueryDate,

current->queryTimes, current->userSatisfiedTimes);

found = 1;

recordingTimes(current,head);

}

current = current->next;

}

if (!found) {

printf("未找到在指定固定时段内的问题记录！\n");

}

}

// 按日期区间段查询函数（辅助函数，供查询功能调用）

void queryByDateInterval(Question \*head, const char \*startDate, const char \*endDate) {

Question \*current = head;

int found = 0;

while (current!= NULL) {

if (strcmp(current->firstQueryDate, startDate) >= 0 && strcmp(current->lastQueryDate, endDate) <= 0) {

printf("问题编号：%d\n问题类型：%d\n问题内容：%s\n问题答案：%s\n首次查询日期：%s\n末次查询日期：%s\n查询次数：%d\n用户满意次数：%d\n",

current->questionId, current->questionType, current->questionContent,

current->questionAnswer, current->firstQueryDate, current->lastQueryDate,

current->queryTimes, current->userSatisfiedTimes);

found = 1;

recordingTimes(current,head);

}

current = current->next;

}

if (!found) {

printf("未找到在指定日期区间内的问题记录！\n");

}

}

// 查询已删除记录函数（辅助函数，供查询功能调用，管理员权限）

void queryDeletedRecords(Question \*head) {

FILE \*file\_undel = fopen("question\_undel.dat", "r");

if (file\_undel == NULL) {

printf("无法打开文件 question\_undel.dat，请检查文件是否存在及权限设置！\n");

return;

}

Question temp;

while (fscanf(file\_undel, "%d %d %s %s %s %s %d %d", &temp.questionId, &temp.questionType,

temp.questionContent, temp.questionAnswer, temp.firstQueryDate,

temp.lastQueryDate, &temp.queryTimes, &temp.userSatisfiedTimes) == 8) {

printf("问题编号：%d\n问题类型：%d\n问题内容：%s\n问题答案：%s\n首次查询日期：%s\n末次查询日期：%s\n查询次数：%d\n用户满意次数：%d\n",

temp.questionId, temp.questionType, temp.questionContent,

temp.questionAnswer, temp.firstQueryDate, temp.lastQueryDate,

temp.queryTimes, temp.userSatisfiedTimes);

}

fclose(file\_undel);

}

// 查询功能函数，根据不同条件查询

void queryQuestions(Question \*head) {

int choice;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("1-----按问题类型查询 2-----按问题内容查询 3-----按查询次数查询\n");

printf("4-----按固定时段查询 5-----按日期区间段查询 \n");

printf("6-----已删除记录查询（管理员才有此项功能） \n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

scanf("%d", &choice);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

switch (choice) {

case 1: {

int questionType;

printf("请输入要查询的问题类型（1 - anyview的常见问题 2 - 智慧树的常见问题\n3 - 编译器的常见问题 4 - 常见语法错误 5 - 常见算法错误）");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

scanf("%d", &questionType);

queryByQuestionType(head, questionType);

break;

}

case 2: {

char content[100];

printf("请输入要查询的问题内容（关键词）：");

scanf("%s", content);

queryByQuestionContent(head, content);

break;

}

case 3: {

int minTimes, maxTimes;

printf("请输入查询次数的最小值和最大值（用空格隔开）：");

scanf("%d %d", &minTimes, &maxTimes);

queryByQueryTimesRange(head, minTimes, maxTimes);

break;

}

case 4: {

char startDate[20], endDate[20];

printf("请输入固定时段的开始日期和结束日期（例如2000-01-01 2000-02-01）：");

scanf("%s %s", startDate, endDate);

queryByFixedPeriod(head, startDate, endDate);

break;

}

case 5: {

char startDate[20], endDate[20];

printf("请输入日期区间段的开始日期和结束日期（例如2000-01-01 2000-02-01）：");

scanf("%s %s", startDate, endDate);

queryByDateInterval(head, startDate, endDate);

break;

}

case 6: {

if (isAdmin) {

queryDeletedRecords(head);

} else {

printf("您无此权限进行已删除记录查询！\n");

}

break;

}

default:

printf("无效的选择，请重新输入！\n");

}

}

// 交换两个节点数据（辅助函数，用于排序）

void swap(Question \*a, Question \*b) {

Question \*p1 = a->next;

Question \*p2 = b->next;

Question temp = \*a;

\*a = \*b;

\*b = temp;

a->next = p1;

b->next = p2;

}

// 按查询次数降序排序函数

void sortByQueryTimes(Question \*head) {

int cnt=1;

Question \*ptr = head;

if (head == NULL) {

return;

}

while (ptr->next!= NULL){

ptr = ptr->next;

cnt++;

}

for(int i=0;i<cnt-1;i++){

ptr = head;

for(int j=0;j<cnt-i-1;j++){

int qt1 = ptr->queryTimes;

int qt2 = ptr->next->queryTimes;

if (qt1 < qt2 || (qt1 == qt2 && ptr->userSatisfiedTimes < ptr->next->userSatisfiedTimes)) {

swap(ptr, ptr->next);

}

ptr = ptr->next;

}

}

}

// 按用户满意程度降序排序函数（辅助函数，用于排序）

void sortByUserSatisfaction(Question \*head) {

int cnt=1;

Question \*ptr = head;

if (head == NULL) {

return;

}

while (ptr->next!= NULL){

ptr = ptr->next;

cnt++;

}

for(int i=0;i<cnt-1;i++){

ptr = head;

for(int j=0;j<cnt-i-1;j++){

double sat1 = (double)ptr->userSatisfiedTimes / (double)ptr->queryTimes;

double sat2 = (double)ptr->next->userSatisfiedTimes / (double)ptr->next->queryTimes;

if (sat1 < sat2 || (sat1 == sat2 && ptr->queryTimes < ptr->next->queryTimes)) {

swap(ptr, ptr->next);

}

ptr = ptr->next;

}

}

}

// 排序功能函数，根据选择调用相应排序逻辑

void sortQuestions(Question \*head) {

int choice;

printf("请选择排序依据（1 - 按查询次数排序，2 - 按用户满意程度排序）：");

scanf("%d", &choice);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

switch (choice) {

case 1:

sortByQueryTimes(head);

break;

case 2:

sortByUserSatisfaction(head);

break;

default:

printf("无效的排序依据选择，请重新输入！\n");

return;

}

displayAllQuestions(head);

}

// 数据录入函数，管理员录入新问题

void inputQuestions(Question \*\*head) {

if (isAdmin) {

int numRecords;

printf("请输入要录入的记录条数：");

scanf("%d", &numRecords);

FILE \*file = fopen("C:\\Users\\AijiA\\Desktop\\24级12班 3124004474 李忠林\\2、源程序及所需文件\\question.txt", "a"); // 以追加模式打开原问题文件

if (file == NULL) {

printf("无法打开文件 question.txt 进行写入，请检查文件权限等问题！\n");

return;

}

for (int i = 0; i < numRecords; i++) {

Question \*newNode = (Question \*)malloc(sizeof(Question));

if (newNode == NULL) {

printf("内存分配失败，录入中断！\n");

fclose(file);

return;

}

printf("请输入问题编号：");

scanf("%d", &newNode->questionId);

printf("请输入问题类型（1 - anyview的常见问题 2 - 智慧树的常见问题 3 - 编译器的常见问题\n4 - 常见语法错误 5 - 常见算法错误）：");

scanf("%d", &newNode->questionType);

while(newNode->questionType>5||newNode->questionType<1){

printf("输入有误！请重新输入：");

scanf("%d", &newNode->questionType);

}

printf("请输入问题内容：");

scanf("%s", newNode->questionContent);

printf("请输入问题答案（最多输入150字）：");

scanf("%s", newNode->questionAnswer);

// 获取当前时间作为首次查询日期和末次查询日期

time\_t t = time(NULL);

struct tm \*tm\_info = localtime(&t);

strftime(newNode->firstQueryDate, sizeof(newNode->firstQueryDate), "%Y-%m-%d", tm\_info);

strcpy(newNode->lastQueryDate, newNode->firstQueryDate);

newNode->queryTimes = 0;

newNode->userSatisfiedTimes = 0;

// 将新录入的数据写入文件

fprintf(file, "%d %d %s %s %s %s %d %d\n", newNode->questionId, newNode->questionType,

newNode->questionContent, newNode->questionAnswer, newNode->firstQueryDate,

newNode->lastQueryDate, newNode->queryTimes, newNode->userSatisfiedTimes);

newNode->next = \*head;

\*head = newNode;

}

fclose(file);

printf("已成功录入！\n");

} else {

printf("您无此权限进行数据录入！\n");

}

}

// 显示所有问题记录函数

void displayAllQuestions(Question \*head) {

Question \*current = head;

while (current!= NULL) {

printf("问题编号：%d\n问题类型：%d\n问题内容：%s\n问题答案：%s\n首次查询日期：%s\n末次查询日期：%s\n查询次数：%d\n用户满意次数：%d\n\n",

current->questionId, current->questionType, current->questionContent,

current->questionAnswer, current->firstQueryDate, current->lastQueryDate,

current->queryTimes, current->userSatisfiedTimes);

current = current->next;

}

}

// 删除指定编号的问题节点（辅助函数，用于数据变更）

void deleteQuestion(Question \*\*head, int questionId) {

Question \*prev = NULL;

Question \*current = \*head;

while (current!= NULL && current->questionId!= questionId) {

prev = current;

current = current->next;

}

if (current == NULL) {

printf("未找到对应问题编号的记录，无法删除！\n");

return;

}

if (prev == NULL) {

\*head = current->next;

} else {

prev->next = current->next;

}

// 将删除的信息保存到另外文件question\_undel.dat

FILE \*file\_undel = fopen("C:\\Users\\AijiA\\Desktop\\24级12班 3124004474 李忠林\\2、源程序及所需文件\\question\_undel.dat", "a");

if (file\_undel == NULL) {

printf("无法打开文件 question\_undel.dat 进行写入，请检查文件权限等问题！\n");

free(current);

return;

}

fprintf(file\_undel, "%d %d %s %s %s %s %d %d\n", current->questionId, current->questionType,

current->questionContent, current->questionAnswer, current->firstQueryDate,

current->lastQueryDate, current->queryTimes, current->userSatisfiedTimes);

fclose(file\_undel);

free(current);

// 更新原文件，去除已删除的记录

FILE \*file = fopen("C:\\Users\\AijiA\\Desktop\\24级12班 3124004474 李忠林\\2、源程序及所需文件\\question.txt", "w");

if (file == NULL) {

perror("无法打开文件 question.txt 进行写入");

free(current);

return;

}

Question \*temp = \*head;

while (temp!= NULL) {

fprintf(file, "%d %d %s %s %s %s %d %d\n", temp->questionId, temp->questionType,

temp->questionContent, temp->questionAnswer, temp->firstQueryDate,

temp->lastQueryDate, temp->queryTimes, temp->userSatisfiedTimes);

temp = temp->next;

}

printf("删除成功！\n");

fclose(file);

//free(current);

}

// 数据变更（更改或删除）函数，管理员操作

void updateQuestions(Question \*\*head) {

if (isAdmin) {

int choice;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("1 - 更改记录 2 - 删除记录\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

scanf("%d", &choice);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

switch (choice) {

case 1: {

int questionId;

printf("请输入要更改的问题编号：");

scanf("%d", &questionId);

Question \*current = \*head;

while (current!= NULL && current->questionId!= questionId) {

current = current->next;

}

if (current == NULL) {

printf("未找到对应问题编号的记录，无法更改！\n");

return;

}

printf("请输入新的问题内容：");

scanf("%s", current->questionContent);

printf("请输入新的问题答案：");

scanf("%s", current->questionAnswer);

// 将更新后的数据写回文件

FILE \*file = fopen("C:\\Users\\AijiA\\Desktop\\24级12班 3124004474 李忠林\\2、源程序及所需文件\\question.txt", "w");

if (file == NULL) {

perror("无法打开文件 question.txt 进行写入");

return;

}

Question \*temp = \*head;

while (temp!= NULL) {

if (fprintf(file, "%d %d %s %s %s %s %d %d\n", temp->questionId, temp->questionType,

temp->questionContent, temp->questionAnswer, temp->firstQueryDate,

temp->lastQueryDate, temp->queryTimes, temp->userSatisfiedTimes) < 0) {

perror("写入 question.txt 文件失败");

break;

}

temp = temp->next;

}

fclose(file);

break;

}

case 2: {

int questionId;

printf("请输入要删除的问题编号：");

scanf("%d", &questionId);

deleteQuestion(head, questionId);

break;

}

default:

printf("无效的选择，请重新输入！\n");

}

} else {

printf("您无此权限进行数据变更！\n");

}

}

// 统计功能函数，区分用户和管理员统计

void statistics(Question \*head) {

if (isAdmin) {

// 管理员统计各个用户在自定义时间段里的求助情况

char startDate[20];

char endDate[20];

printf("请输入管理员统计的开始日期和结束日期（用空格隔开）：");

scanf("%s %s", startDate, endDate);

Question \*current = head;

int totalQueryTimes = 0;

while (current!= NULL) {

if (strcmp(current->firstQueryDate, startDate) >= 0 && strcmp(current->lastQueryDate, endDate) <= 0) {

totalQueryTimes += current->queryTimes;

}

current = current->next;

}

printf("在管理员指定自定义时间段内的总查询次数：%d\n", totalQueryTimes);

} else {

// 用户统计自己在一个自定义时间段里的求助情况

char startDate[20];

char endDate[20];

printf("请输入用户统计的开始日期和结束日期（用空格隔开）：");

scanf("%s %s", startDate, endDate);

Question \*current = head;

int totalQueryTimes = 0;

while (current!= NULL) {

if (strcmp(current->firstQueryDate, startDate) >= 0 && strcmp(current->lastQueryDate, endDate) <= 0) {

totalQueryTimes += current->queryTimes;

}

current = current->next;

}

printf("在用户指定自定义时间段内的总查询次数：%d\n", totalQueryTimes);

}

}

// 判断是否为管理员权限的函数

void isAdm() {

char password[]="123abc";

char yesSkip[]="是";

char notSkip[]="否";

char str[7];

char prev\_isSkip[3];

char curr\_isSkip[3];

printf("是否进行管理员登录（是或否）：");

scanf("%s",prev\_isSkip);

if(strcmp(yesSkip, prev\_isSkip) != 0 && strcmp(notSkip, prev\_isSkip) != 0){

printf("输入错误！请重新输入（是或否）：");

scanf("%s",prev\_isSkip);

}

if(strcmp(yesSkip, prev\_isSkip) == 0){

printf("请输入密码：");

scanf("%s",str);

if(strcmp(password, str) == 0){

printf("登陆成功！您已成为管理员！\n");

isAdmin=1;

}else{

printf("密码错误！是否重新登录:");

scanf("%s",curr\_isSkip);

if(strcmp(yesSkip, curr\_isSkip) == 0){

printf("请输入密码：");

scanf("%s",str);

if(strcmp(password, str) == 0){

printf("登陆成功！您已成为管理员！\n");

isAdmin=1;

}else{

printf("登陆失败！已默认您为用户！\n");

}

}

}

}

}