

XI`AN TECHNOLOGICAL UNIVERSITY

课程设计报告

课程名称：

专 业：

班 级：

姓 名：

学 号：

指导教师： 刘敏侠

成 绩：

2025 年 6 月 20 日

1. 问题描述

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | C语言项目实践 | | | | | 指导教师 | | 刘敏侠 | |
| 课设题目1 | 学生信息管理系统 | | | | | | | |  |
| 学号 |  | 姓名 |  | 地点 |  | | 完成时间 | | 2025/6/16  2025/6/20 |
| 设计任务 | 建立一个学生信息管理系统，利用单链表实现，具有如下的五大功能模块，具体要求如下：  qq_pic_merged_1687695704408  学生信息管理系统 | | | | | | | | |
| 设计要求 | 1. 输入记录模块   逐个输入学生基本信息，学生信息包含：学号、姓名、性别、年龄、班级等。   1. 查询记录模块   按照学生的学号或者姓名进行查询，把满足条件的学生信息显示出来。   1. 更新记录模块   修改某一个学生的信息，插入一个学生信息；删除一个学生的信息。   1. 统计记录模块   能够按班级、性别统计学生数量，并输出统计结果；。   1. 输出记录模块   显示所有学生的信息。   1. 能够将所有学生信息保存到文件中，并能从文件中读取学生信息；   以上为最基本要求，可根据自己的理解在此基础上对各个模块增加和完善相应的功能，使得整个系统能更加接近实际的应用。 | | | | | | | | |
| 成果要求 | 1. 设计出完整流程图。  2．编写出全部模块对应的函数。  3．在集成开发工具中调试通过。  4．参加答辩。  5．写出完整、规范的课程设计报告，在指定的时间内提交指导教师。 | | | | | | | | |
| 设计报告  要求 | 1．封面：（格式附后）  2．课程设计任务书  3．课程设计报告：  ①题目用黑体三号，段后距18磅（或1行），居中对齐；  ②标题用黑体四号，段前、段后距6磅（或0.3行）；  ③正文用小四号宋体，行距为1.25倍行距；  ④标题按“一”、“㈠”、“1”、“⑴”顺序编号。  ⑤报告和软件打包以学号姓名命名，邮件主题也为学号姓名。  ⑴ 系统总体方案  ①题目用黑体三号，段后距18磅（或1行），居中对齐；  ②标题用黑体四号，段前、段后距6磅（或0.3行）；  ③正文用小四号宋体，行距为1.25倍行距；  ④标题按“一”、“㈠”、“1”、“⑴”顺序编号。  ⑤报告和软件打包以学号姓名命名，邮件主题也为学号姓名。  ①题目用黑体三号，段后距18磅（或1行），居中对齐；  ②标题用黑体四号，段前、段后距6磅（或0.3行）；  ③正文用小四号宋体，行距为1.25倍行距；  ④标题按“一”、“㈠”、“1”、“⑴”顺序编号。  ⑤报告和软件打包以学号姓名命名，邮件主题也为学号姓名。  ⑵ 设计思路和主要步骤  ⑶ 各功能模块和流程图  ⑷ 设计代码  ⑸ 心得体会 | | | | | | | | |

1. 算法设计--主控模块与用户交互设计

### ****实验报告：主控模块与用户交互设计****

**负责代码**: main.cpp **核心任务**: 深入分析和阐述程序的“骨架”——主控模块的设计思想、实现细节与优化过程。报告重点在于用户体验、程序流程控制和模块化调用。

### ****第一章：绪论****

#### ****1.1 项目背景与意义****

在信息化浪潮席卷全球的今天，教育领域的数字化转型已成为不可逆转的趋势。传统的纸质化学生信息管理方式，如纸质档案、手动填写的表格等，日益暴露出其固有的弊端。这些方式不仅效率低下、人力成本高昂，而且在数据查询、更新、备份和统计分析等方面存在着巨大的困难。纸质档案易于丢失、损坏，数据检索耗时耗力，且无法进行快速、复杂的数据聚合与分析，这极大地限制了教育管理的现代化水平。

因此，开发一套功能完善、操作便捷的电子化学生信息管理系统，是解决上述痛症、顺应时代发展的必然要求。本“学生信息管理系统”项目，旨在利用C语言，构建一个能够实现对学生基本信息（学号、姓名、班级等）进行集中式、数字化管理的控制台应用程序。该系统的成功实现，不仅能显著提升信息管理的效率与准确性，使管理员从繁琐的手工劳动中解放出来，更能为学校的教学决策、学生工作提供及时、有效的数据支持，具有重要的现实意义和应用价值。本项目也是对C语言编程基础、数据结构、模块化编程思想的一次综合性实践与检验。

#### ****1.2 本模块（主控模块）的设计目标****

main.cpp 文件作为本系统的入口和总指挥中心，其设计好坏直接决定了整个程序的稳定性、健壮性和用户体验。它不直接参与具体的业务逻辑运算，但其承担的“承上启下”的关键作用，使其设计目标必须清晰明确。本模块的核心设计目标可细化为以下四点：

**提供一个清晰、易于理解的用户操作界面。** 用户交互（Human-Computer Interaction, HCI）是软件的“门面”。主控模块的首要目标是为用户呈现一个直观、无歧义的菜单界面。所有功能选项应清晰列出，引导用户进行正确的操作，最大限度地降低用户的学习成本和误操作的可能性，从而提供流畅、友好的用户体验。

**构建一个稳定、健SH壮的程序主循环。** 程序的主循环是整个应用的“心脏”。它必须足够稳定，能够持续接收用户的指令并做出响应，而不是执行一次操作后就退出。同时，它必须具备基本的容错能力，即当用户输入非法指令（如非菜单选项的数字或字符）时，程序不应崩溃，而应能给出清晰的错误提示，并引导用户重新输入，保证程序的健壮性。

**实现对业务逻辑模块的正确、高效调用。** 遵循“高内聚，低耦合”的软件工程思想，主控模块（main.cpp）作为“表现层”，应与“业务逻辑层”（student.cpp 中实现的具体功能）进行有效分离。本模块的目标是充当一个高效的“调度器”，根据用户的选择，精确地调用在 student.h 中声明的、在 student.cpp 中实现的对应功能函数，并将必要的数据（如链表头指针 head）传递给它们，实现各模块间的协同工作。

**确保程序生命周期内资源的有效管理。** 一个设计精良的程序必须妥善管理其整个生命周期。本模块的设计目标涵盖了程序的启动与终止两个关键节点。在程序启动时，必须负责调用 loadFromFile 函数，从磁盘恢复上一次的工作状态，完成程序的初始化；在程序即将终止时，必须负责调用 freeList 函数，确保所有动态分配的内存（链表节点）得到完全释放，避免内存泄漏，保证程序的洁净、安全退出。

#### ****1.3 开发环境****

* **操作系统**: Windows 10 / Windows 11
* **集成开发环境 (IDE)**: Visual Studio 2022
* **编译器**: Microsoft Visual C++ (MSVC)
* **核心语言**: C语言 (C89/C99标准)

### ****第二章：主控模块总体设计****

#### ****2.1 程序总体架构****

本系统在设计上遵循了经典的三层架构思想，main.cpp 在其中扮演着承上启下的关键角色。我们可以将程序的结构理解为：

**表现层 (Presentation Layer)**:

**核心文件**: main.cpp

**职责**: 该层直接面向用户，是用户与系统交互的唯一入口。它负责显示菜单 (showMenu)、接收并解析用户的输入、向用户反馈操作结果（如“添加成功”、“未找到学生”等提示信息）。它不关心学生信息是如何存储或操作的，只负责调用下层提供的服务。

**业务逻辑层 (Business Logic Layer)**:

**核心文件**: student.cpp

**职责**: 该层是系统的核心，负责处理所有的具体业务。它实现了对学生数据进行增、删、改、查、统计等所有实际操作。它接收来自表现层的指令和数据，完成处理后将结果返回。

**数据接口层 (Data Interface Layer)**:

**核心文件**: student.h

**职责**: 该层定义了系统的数据结构（Student, Node）和各模块之间的“通信契约”（函数原型声明）。main.cpp 和 student.cpp 都包含此头文件，确保了它们对数据结构和函数接口有统一的认知，实现了表现层与业务逻辑层的解耦。

**交互流程示例（以“录入学生信息”为例）：** 用户在main.cpp驱动的菜单中选择“1” -> main.cpp的switch语句捕获该选择，调用addStudent(head)函数 -> 该调用通过student.h中声明的函数原型，定位到student.cpp中addStudent函数的具体实现 -> student.cpp执行完毕后，控制权返回main.cpp -> main.cpp接着调用saveToFile(head, FILENAME)进行数据持久化 -> 流程结束，等待下一次用户输入。

#### ****2.2 主函数**** main() ****流程****

1. **[开始]** 程序启动。
2. **[初始化]** 调用 Node\* head = loadFromFile(FILENAME);，尝试从 "student.txt" 文件加载数据来初始化链表。如果文件不存在，则创建一个空的带头节点的链表。head 指针将管理整个链表。
3. **[进入主循环]** do { ... } while (choice != 0); 结构开始执行。
4. **[显示菜单]** 在循环体内部，首先调用 showMenu();，在控制台打印出美观、清晰的用户操作菜单。
5. **[获取用户输入]** 执行 scanf("%d", &choice);，等待用户输入一个代表其选择的整数。
6. **[清理输入缓冲区]** 立即调用 clearInputBuffer();，消耗掉用户输入数字后残留在标准输入流中的换行符或其他字符，这是避免对后续输入函数（特别是 getchar）产生干扰的关键一步。
7. **[进入调度器]** switch (choice) 结构开始判断。
8. **[分支处理]**
   * 若 choice 为 1 或 3，则调用对应的业务函数（addStudent 或 updateMenu），之后 **立即调用** saveToFile() 将变更持久化到磁盘。
   * 若 choice 为 2, 4, 5，则仅调用对应的业务函数 (queryMenu, statisticsMenu, displayAllStudents)。
   * 若 choice 为 0，则打印退出信息。
   * 若 choice 为其他任何值 (default)，则打印“无效输入”的错误提示。
9. **[流程暂停]** switch 结束后，执行 printf("\n按 Enter 键继续..."); 和 getchar();，暂停程序，等待用户阅读屏幕上的输出信息，用户按回车后程序才继续。
10. **[清屏]** 执行 system("cls");，清空控制台屏幕，为下一次菜单的显示提供一个干净的界面。
11. **[循环判断]** 到达 do-while 的尾部，判断 choice != 0 是否成立。
    * 如果为 true（用户未选择退出），则程序跳转回 **步骤4**，开始下一次循环。
    * 如果为 false（用户选择了 0），则循环终止。
12. **[资源释放]** 退出循环后，执行 freeList(head);，遍历并释放整个链表占用的所有动态内存。
13. **[程序终止]** 执行 return 0;，程序正常结束。
14. **[结束]**

#### ****2.3 用户交互设计 (****showMenu****)****

##### ****2.3.1 菜单设计原则****

showMenu 函数虽然代码简单，但其背后蕴含着重要的人机交互设计原则：

**简洁性 (Simplicity)**: 菜单选项直截了当，没有使用任何晦涩或有歧义的词语。功能被明确地编号，用户可以轻松地将数字选项与具体功能对应起来，避免了信息过载。

**一致性 (Consistency)**: 整个程序的交互核心就是这个菜单。每次操作结束后，都会清屏并重新显示这个结构完全一致的菜单，这种可预期的行为模式为用户提供了稳定感和控制感。

**易懂性 (Clarity)**: 菜单使用了 ========= 和 || 字符构建了一个视觉上的边框，这在纯文本环境中有效地将菜单内容与程序的其他输出（如查询结果、错误提示）区隔开来，形成了一个视觉焦点，引导用户专注于菜单本身。

##### ****2.3.2 界面布局分析****

showMenu 函数通过一系列 printf 语句，精心构建了一个文本用户界面（Text-based User Interface, TUI）。这种设计的考量在于：

**视觉分隔**: 使用 ========= 作为顶部和底部的分隔线，|| 作为两侧的边框，创建了一个“窗口”或“盒子”的视觉效果。这在单调的控制台环境中极大地提升了信息的可读性。

**对齐与留白**: printf 语句中的文本内容通过空格进行了手动对齐，并在功能列表的上下留出了空白的 || 行。这种留白设计避免了内容的拥挤感，使得界面看起来更加专业和整洁。

##### ****2.3.3 交互模式探讨****

本项目采用了经典的 **菜单驱动界面 (Menu-Driven Interface, MDI)**。这种模式的优点和适用性在于：

**优点**:

**易于学习**: 用户无需记忆任何命令，所有可用功能都直观地展示在屏幕上。对于初次使用者或不熟悉计算机的用户来说非常友好。

**减少输入错误**: 用户只需输入数字编号，相比于输入完整的命令字符串，大大降低了拼写错误的可能性。

**其他交互模式（对比）**:

**命令行界面 (Command-Line Interface, CLI)**: 用户通过输入命令和参数来执行操作（例如 add --name "张三" --id "001"）。

**优点**: 对于熟练用户来说操作效率极高，且便于通过脚本实现自动化操作。

**缺点**: 学习曲线陡峭，用户必须记忆大量命令和参数格式。

**图形用户界面 (Graphical User Interface, GUI)**: 使用窗口、按钮、文本框等图形元素进行交互。

**优点**: 最为直观、现代，符合大多数用户的使用习惯。

**缺点**: 开发复杂度远高于控制台程序，需要依赖特定的图形库（如Qt, GTK, MFC等），超出了本项目C语言基础实践的范畴。

**结论**: 考虑到本项目的定位是一个基础的、教学性质的管理系统，其目标用户是普通的操作员而非高级程序员。因此，选择菜单驱动界面是最为合适的方案，它在开发复杂度和用户友好度之间取得了最佳的平衡。

### ****第三章：****main.cpp ****详细实现与代码分析****

#### ****3.1 头文件与宏定义****

#include "student.h": 这行代码是main.cpp与系统其他部分沟通的桥梁。它起到了“契约”的作用，main.cpp通过它得知存在哪些可以调用的函数（如addStudent, loadFromFile等）以及它们需要什么参数、会返回什么结果。main.cpp本身并不关心这些函数内部复杂的实现逻辑，它只需要知道这个“接口”的定义即可。这种“接口与实现分离”的原则是模块化编程的核心，极大地降低了代码模块间的耦合度，使得分工协作和后期维护成为可能。

#define FILENAME "student.txt": 此处使用宏定义来指定数据文件的名称。这样做避免了在代码中多次直接写入"student.txt"这个“魔法字符串”（Magic String）。其好处是显而易见的：当需要更改文件名时（例如，在发布版本中改为student.dat），只需修改#define这一行即可，无需在代码中进行多处查找和替换。这大大提高了代码的可维护性和可读性。

#### ****3.2 程序初始化 (****loadFromFile****)****

main函数的第一行有效代码 Node\* head = loadFromFile(FILENAME); 是整个程序能够“记忆”的关键。如果缺少这一步，那么程序每次启动时，链表都是空的，用户之前的所有录入工作都将丢失。此调用确保了程序的 **状态持久性**，它将上一次会话结束时保存在硬盘文件中的数据重新加载到内存的链表结构中，恢复了程序的工作状态，为用户提供了连续、无缝的操作体验。

#### ****3.3 主循环 (****do-while****) 的设计****

**为何使用** do-while: 选择 do-while 循环而非 while 循环，是因为 do-while 是一个 **“后测试”** 循环。它的循环体（显示菜单、获取输入等）至少会无条件执行一次，然后才在循环尾部检查循环条件（choice != 0）。这完美地契合了我们的需求：程序启动后，我们希望首先向用户展示菜单，然后再根据用户的选择决定是否继续循环。如果使用 while 循环，就必须在循环前对choice变量进行一次初始化，代码会稍显冗余。

**循环条件** choice != 0: 将 0 作为退出程序的特定指令，是一种广泛接受的、符合用户直觉的交互设计惯例。在许多应用程序中，0 或 q (quit) 都被用作退出命令。这个清晰的循环终止条件使得程序流程的控制逻辑变得简单明了。

#### ****3.4 核心调度器 (****switch-case****)****

switch-case 结构是main函数的 **“命令分发中心”** 或 **“中央调度器”**。它接收用户的整数选择 choice，并像一个交通警察一样，将程序的执行流程引导到正确的业务逻辑函数上。

case 1 **至** case 5: 每个 case 对应一个菜单选项，其内部是对相应功能函数的直接调用。这种结构清晰地将“用户意图”映射到了“程序行为”上。

**即时保存策略 (**case 1 **和** case 3**)**: 在执行了 addStudent (添加) 和 updateMenu (更新，包含修改和删除) 这两个会改变数据的操作后，程序立即调用了 saveToFile。这是一种 **“即时保存”** 的数据安全策略。其优点是最大程度地保证了数据的安全性，即使用户的电脑突然断电或程序意外崩溃，刚刚完成的修改也已经被写入了磁盘。缺点是在处理海量数据时，频繁的磁盘I/O操作可能会带来性能开销。但对于本项目这种规模的应用来说，安全性是首要考虑的。

**退出逻辑的潜在问题与优化 (**case 0**)**:

**当前实现**: case 0: { FILE\* fp = fopen(FILENAME, "w"); } ...

**问题分析**: 这行代码的意图可能是清空文件，但它存在两个问题：1. **资源未关闭**: fopen 返回的文件指针 fp 从未被 fclose 关闭，这在严格的编程规范中属于资源泄漏。虽然程序即将退出，操作系统会回收资源，但这仍是一个不良的编程习惯。

2. **逻辑不清晰**: 退出时为何要清空文件？这段代码的意图并不明确，且没有给用户任何提示。

**改进方案**: 更好的做法是移除这行代码，或者提供一个更友好的退出机制，例如询问用户是否需要保存最后的更改，或者在退出时自动保存一次。一个更健壮的退出逻辑应该是：

#### ****3.5 用户输入处理****

scanf("%d", &choice);: scanf 是C语言中用于格式化输入的标准函数。它会根据 %d 格式说明符，尝试从标准输入流中读取一个十进制整数并存入 choice 变量。然而，scanf 的一个著名“陷阱”是，它在成功读取数字后，会将输入缓冲区中的换行符 \n（用户按回车键产生）保留下来。此外，如果用户输入的不是数字（例如一个字母），scanf 会读取失败，该字母会一直留在缓冲区中，可能导致程序陷入死循环。

clearInputBuffer();: 这个自定义的辅助函数是解决上述 scanf 陷阱的“神器”。它在 scanf 执行后立即被调用，其内部的 while ((c = getchar()) != '\n' && c != EOF); 循环会不断地从输入缓冲区中读取并丢弃字符，直到遇到换行符 \n 或文件结束符 EOF 为止。通过这种方式，它“吃掉”了所有残留的垃圾字符，确保了输入缓冲区的“洁净”，从而保证了后续依赖输入的函数（尤其是 getchar()）能够正常工作。

#### ****3.6 程序暂停与清屏****

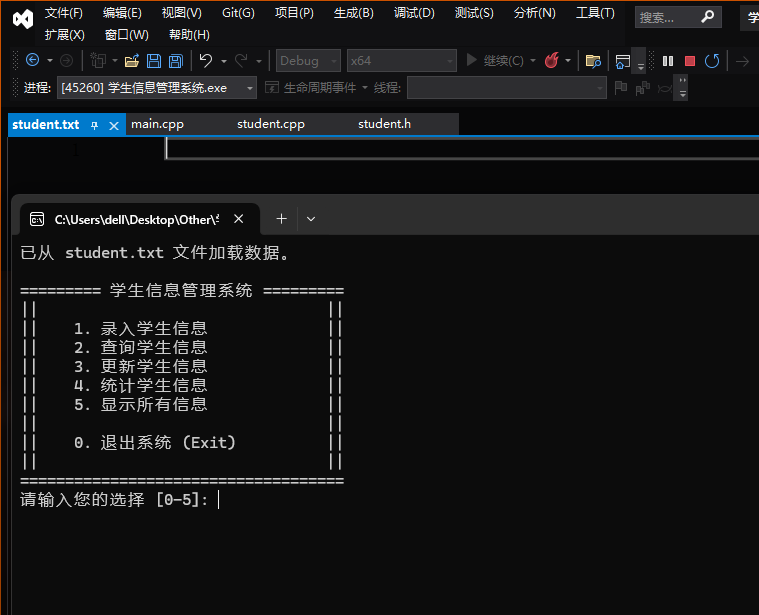
getchar();: 此函数的作用是从标准输入读取一个字符。在printf("\n按 Enter 键继续...");之后调用它，程序会在此处暂停，等待用户输入。由于之前调用了clearInputBuffer，缓冲区是干净的，所以getchar会等待用户按下键盘上的任意键（通常是回车键），从而达到了暂停程序、让用户有充足时间阅读屏幕信息的效果。这是提升控制台程序用户体验的一个简单而有效的方法。

system("cls");: system() 函数可以执行一个操作系统命令。"cls" 是Windows系统中用于清空命令提示符窗口内容的命令。在每次循环的末尾调用它，可以在下一次显示主菜单前清空屏幕，避免了信息的滚动和混乱，使得界面始终保持整洁。需要注意的是，该命令不具备跨平台性，在Linux或macOS系统中，清屏命令是 "clear"。一个更健- robust 的程序会使用条件编译来处理这种差异：

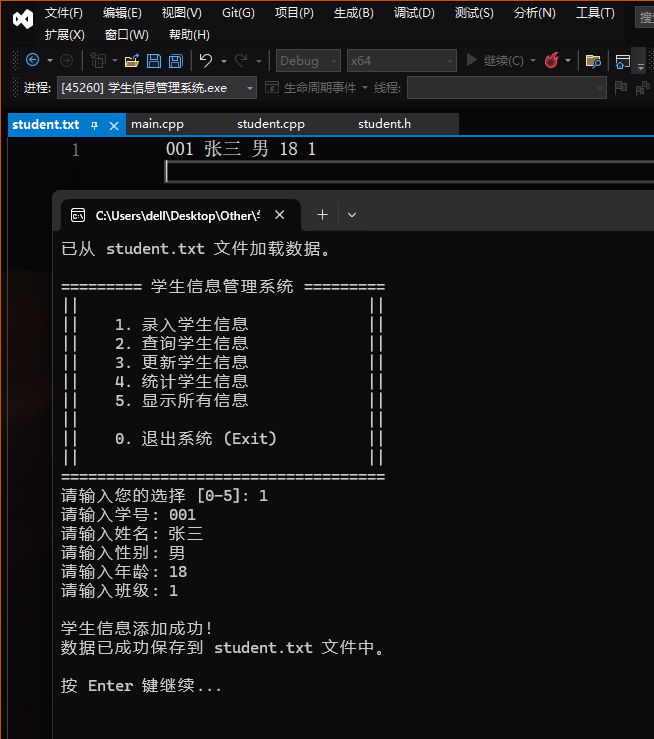
#### ****3.7 资源释放 (****freeList****)****

在 main 函数的 return 0; 之前调用 freeList(head); 是整个程序中最能体现严谨编程思想的关键步骤之一。在程序运行过程中，每当添加一个学生信息，我们都通过 malloc 向操作系统申请了一块内存来存放 Node 节点。操作系统并不会自动回收这些内存。如果程序在退出前不显式地调用 free 函数将它们一一归还，就会造成 **内存泄漏 (Memory Leak)**。对于一个短暂运行的控制台应用，其影响可能不明显，但对于需要7x24小时运行的服务器程序而言，持续的内存泄漏最终会导致系统资源耗尽而崩溃。因此，freeList 函数遍历整个链表，逐个释放所有节点的内存，是“有借有还，再借不难”的良好编程实践，是构建稳定、可靠软件的基石。

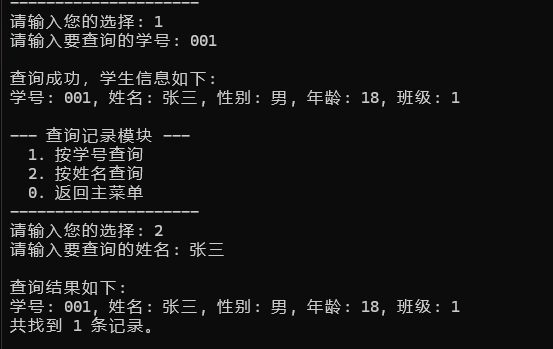
1. 算法实现
2. 程序主页面，启动与加载：程序启动后，首先显示从 student.txt 文件加载数据的提示，然后展示主菜单，等待用户操作 。



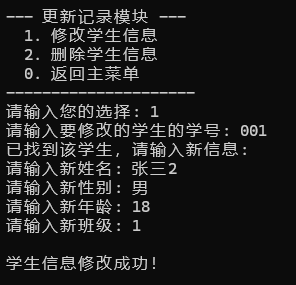
1. 录入学生信息，信息录入：下图展示了录入一个学号为"001"的学生信息的完整过程。添加成功后，系统提示数据已保存到文件 。

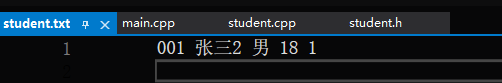


1. 查询学生信息，信息查询：系统提供了按学号和姓名两种查询方式。下图演示了分别使用这两种方式查询名为“张三”的学生，并准确返回了结果 。

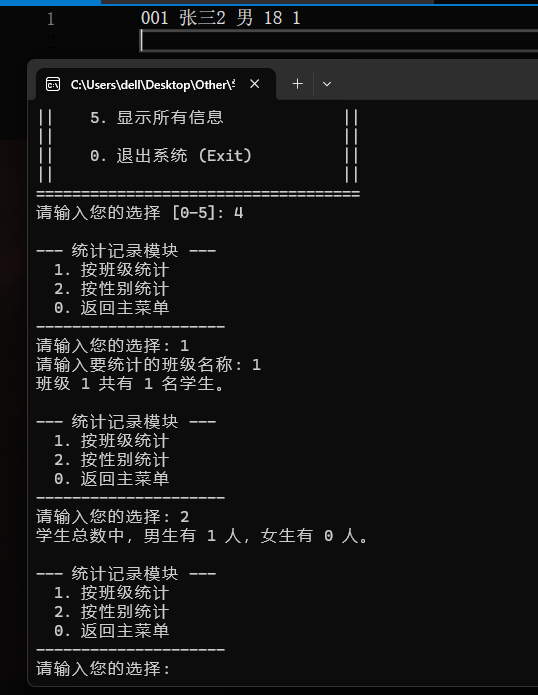


1. 更新学生信息，信息更新：用户可以修改或删除学生信息。下图展示了将学号"001"的学生的姓名从“张三”修改为“张三2”的操作。修改成功后，student.txt 文件中的内容也同步更新





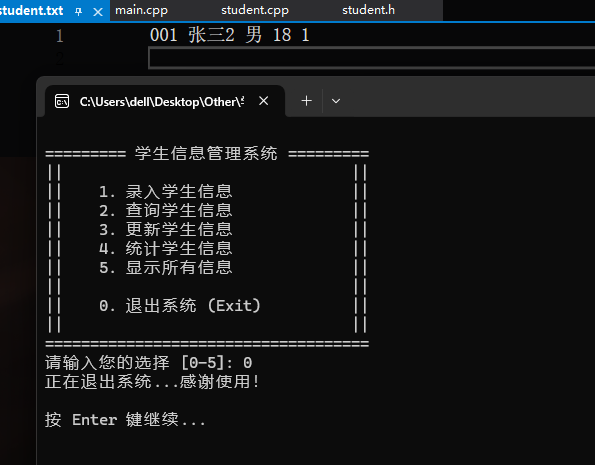
1. 统计学生信息，信息统计与显示：系统能够按班级和性别进行统计，也可以完整地列出所有学生的信息，方便用户全局浏览 。

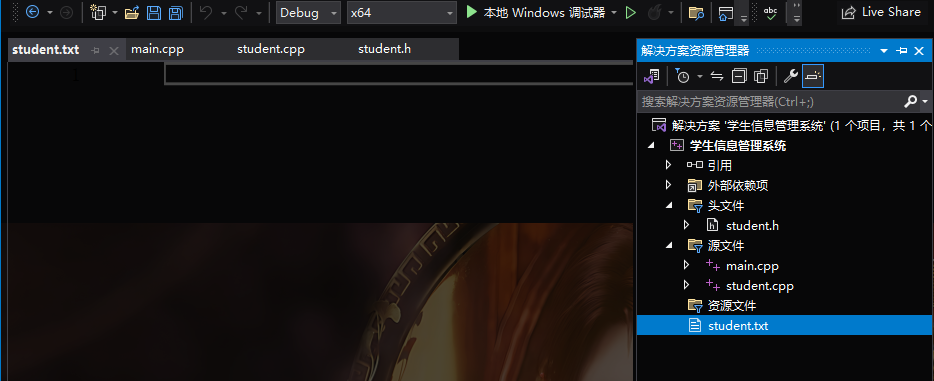


1. 显示所有信息



1. 退出系统 Exit





四．心得体会

通过本次C语言课程设计，我不仅加深了对C语言基础和链表数据结构的理解，更重要的是，我成功地为系统添加了文件操作功能，实现了数据的持久化存储。这个过程让我收获颇丰。

首先，在实现文件读写功能时，我深刻体会到了数据格式统一的重要性。在 saveToFile 函数中，我使用 fprintf 以“字符串 空格 字符串 空格 ...”的格式写入数据，那么在 loadFromFile 函数中，就必须使用与之完全匹配的 fscanf 格式字符串来读取，任何一个字段的顺序或类型不匹配，都会导致数据读取混乱甚至程序崩溃。这让我明白了接口和协议在编程中的重要性。

其次，我学会了处理文件操作中的各种边界情况。比如，程序第一次运行时，数据文件 students.dat 是不存在的，fopen 会返回 NULL。我必须在代码中正确处理这种情况，不是报错退出，而是将其视为一个正常的初始状态，返回一个空链表让系统继续运行。这培养了我编写健壮（Robust）程序的意识。

此外，将加载功能放在程序启动时自动执行，将保存功能作为菜单选项供用户手动触发，是一种经典且用户友好的设计模式。我还额外添加了在退出时提醒用户保存的功能，这让程序的人性化体验又上了一个台阶。

总而言之，这次对文件操作的实践，让我真正将C语言的知识应用到了一个“看得见、摸得着”的实际问题上。它不再是冷冰冰的链表节点，而是可以被保存、被恢复的学生记录。这次经历极大地提升了我的自信心，也让我对软件开发的完整流程——从内存操作到磁盘存储——有了更全面的认识。

五．参考文献

[1]张学孟,梅诗冬,吕家威.基于在线信息化平台的学生信息管理系统运用研究[J].信息系统工程,2025,(06):8-11.

[1]王英.C语言中循环转递归函数策略研究[J].科学技术创新,2025,(15):71-74.

[1]单宝华,佟智慧.C语言在线编程代码缺陷检测方法研究[J].网络安全技术与应用,2025,(05):56-59.

[1]夏黎晨.编程语言中的深拷贝和浅拷贝对算法设计的影响研究[J].软件,2025,46(04):151-153.

[1]葛方振,洪留荣.C语言程序设计基础实验教程[M].中国铁道出版社:202212.108.

源代码

1. student.h (头文件 / 链接口)
2. #ifndef STUDENT\_H
3. #define STUDENT\_H
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include <string.h>
7. *// 学生信息结构体*
8. typedef struct Student {
9. char id[20];
10. char name[50];
11. char gender[10];
12. int age;
13. char className[50];
14. } Student;
15. *// 链表节点结构体*
16. typedef struct Node {
17. Student data;
18. struct Node\* next;
19. } Node;
20. *// === 函数原型声明 (模块接口) ===*
21. *// 1. 输入模块*
22. void addStudent(Node\* head);
23. *// 2. 查询模块*
24. void queryMenu(Node\* head);
25. *// 3. 更新模块*
26. void updateMenu(Node\* head);
27. *// 4. 统计模块*
28. void statisticsMenu(Node\* head);
29. *// 5. 输出模块*
30. void displayAllStudents(Node\* head);
31. *// 6. 文件操作模块 (数据持久化)*
32. void saveToFile(Node\* head, const char\* filename); *// 保存数据到文件*
33. Node\* loadFromFile(const char\* filename);         *// 从文件加载数据*
34. *// 辅助函数*
35. void clearInputBuffer();
36. void freeList(Node\* head);
37. #endif *// STUDENT\_H*
38. student.cpp (函数实现文件)
39. #define \_CRT\_SECURE\_NO\_WARNINGS
40. #include "student.h"
41. *// 清空输入缓冲区*
42. void clearInputBuffer() {
43. int c;
44. while ((c = getchar()) != '\n' && c != EOF);
45. }
46. *// 检查学号是否唯一*
47. int isIdUnique(Node\* head, const char\* id) {
48. Node\* p = head->next;
49. while (p != NULL) {
50. if (strcmp(p->data.id, id) == 0) {
51. return 0; *// 不唯一*
52. }
53. p = p->next;
54. }
55. return 1; *// 唯一*
56. }
57. *// 1. 输入记录模块*
58. void addStudent(Node\* head) {
59. Node\* newNode = (Node\*)malloc(sizeof(Node));
60. if (newNode == NULL) {
61. printf("内存分配失败！\n");
62. return;
63. }
64. printf("请输入学号: ");
65. scanf("%s", newNode->data.id);
66. clearInputBuffer();
67. if (!isIdUnique(head, newNode->data.id)) {
68. printf("错误：该学号已存在！\n");
69. free(newNode);
70. return;
71. }
72. printf("请输入姓名: ");
73. scanf("%s", newNode->data.name);
74. clearInputBuffer();
75. printf("请输入性别: ");
76. scanf("%s", newNode->data.gender);
77. clearInputBuffer();
78. printf("请输入年龄: ");
79. scanf("%d", &newNode->data.age);
80. clearInputBuffer();
81. printf("请输入班级: ");
82. scanf("%s", newNode->data.className);
83. clearInputBuffer();
84. newNode->next = NULL;
85. *// 找到链表尾部并插入*
86. Node\* p = head;
87. while (p->next != NULL) {
88. p = p->next;
89. }
90. p->next = newNode;
91. printf("\n学生信息添加成功！\n");
92. }
93. *// 2. 查询记录模块*
94. void searchStudentById(Node\* head) {
95. if (head->next == NULL) {
96. printf("当前无任何学生信息。\n");
97. return;
98. }
99. char id[20];
100. printf("请输入要查询的学号: ");
101. scanf("%s", id);
102. clearInputBuffer();
103. Node\* p = head->next;
104. int found = 0;
105. while (p != NULL) {
106. if (strcmp(p->data.id, id) == 0) {
107. printf("\n查询成功，学生信息如下:\n");
108. printf("学号: %s, 姓名: %s, 性别: %s, 年龄: %d, 班级: %s\n",
109. p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);
110. found = 1;
111. break;
112. }
113. p = p->next;
114. }
115. if (!found) {
116. printf("未找到学号为 %s 的学生。\n", id);
117. }
118. }
119. void searchStudentByName(Node\* head) {
120. if (head->next == NULL) {
121. printf("当前无任何学生信息。\n");
122. return;
123. }
124. char name[50];
125. printf("请输入要查询的姓名: ");
126. scanf("%s", name);
127. clearInputBuffer();
128. Node\* p = head->next;
129. int found = 0;
130. printf("\n查询结果如下:\n");
131. while (p != NULL) {
132. if (strcmp(p->data.name, name) == 0) {
133. printf("学号: %s, 姓名: %s, 性别: %s, 年龄: %d, 班级: %s\n",
134. p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);
135. found++;
136. }
137. p = p->next;
138. }
139. if (found == 0) {
140. printf("未找到姓名为 %s 的学生。\n", name);
141. }
142. else {
143. printf("共找到 %d 条记录。\n", found);
144. }
145. }
146. void queryMenu(Node\* head) {
147. int choice;
148. do {
149. printf("\n--- 查询记录模块 ---\n");
150. printf("  1. 按学号查询\n");
151. printf("  2. 按姓名查询\n");
152. printf("  0. 返回主菜单\n");
153. printf("---------------------\n");
154. printf("请输入您的选择: ");
155. scanf("%d", &choice);
156. clearInputBuffer();
157. switch (choice) {
158. case 1: searchStudentById(head); break;
159. case 2: searchStudentByName(head); break;
160. case 0: break;
161. default: printf("无效选择，请重新输入。\n");
162. }
163. } while (choice != 0);
164. }
165. *// 3. 更新记录模块*
166. void modifyStudent(Node\* head) {
167. if (head->next == NULL) {
168. printf("当前无任何学生信息。\n");
169. return;
170. }
171. char id[20];
172. printf("请输入要修改的学生的学号: ");
173. scanf("%s", id);
174. clearInputBuffer();
175. Node\* p = head->next;
176. int found = 0;
177. while (p != NULL) {
178. if (strcmp(p->data.id, id) == 0) {
179. printf("已找到该学生，请输入新信息:\n");
180. printf("请输入新姓名: ");
181. scanf("%s", p->data.name);
182. clearInputBuffer();
183. printf("请输入新性别: ");
184. scanf("%s", p->data.gender);
185. clearInputBuffer();
186. printf("请输入新年龄: ");
187. scanf("%d", &p->data.age);
188. clearInputBuffer();
189. printf("请输入新班级: ");
190. scanf("%s", p->data.className);
191. clearInputBuffer();
192. printf("\n学生信息修改成功！\n");
193. found = 1;
194. break;
195. }
196. p = p->next;
197. }
198. if (!found) {
199. printf("未找到学号为 %s 的学生。\n", id);
200. }
201. }
202. void deleteStudent(Node\* head) {
203. if (head->next == NULL) {
204. printf("当前无任何学生信息。\n");
205. return;
206. }
207. char id[20];
208. printf("请输入要删除的学生的学号: ");
209. scanf("%s", id);
210. clearInputBuffer();
211. Node\* prev = head;
212. Node\* curr = head->next;
213. int found = 0;
214. while (curr != NULL) {
215. if (strcmp(curr->data.id, id) == 0) {
216. prev->next = curr->next;
217. free(curr);
218. printf("学号为 %s 的学生信息已删除。\n", id);
219. found = 1;
220. break;
221. }
222. prev = curr;
223. curr = curr->next;
224. }
225. if (!found) {
226. printf("未找到学号为 %s 的学生。\n", id);
227. }
228. }
229. void updateMenu(Node\* head) {
230. int choice;
231. do {
232. printf("\n--- 更新记录模块 ---\n");
233. printf("  1. 修改学生信息\n");
234. printf("  2. 删除学生信息\n");
235. printf("  0. 返回主菜单\n");
236. printf("---------------------\n");
237. printf("请输入您的选择: ");
238. scanf("%d", &choice);
239. clearInputBuffer();
240. switch (choice) {
241. case 1: modifyStudent(head); break;
242. case 2: deleteStudent(head); break;
243. case 0: break;
244. default: printf("无效选择，请重新输入。\n");
245. }
246. } while (choice != 0);
247. }
248. *// 4. 统计记录模块*
249. void statisticsByClass(Node\* head) {
250. if (head->next == NULL) {
251. printf("当前无任何学生信息。\n");
252. return;
253. }
254. char className[50];
255. printf("请输入要统计的班级名称: ");
256. scanf("%s", className);
257. clearInputBuffer();
258. Node\* p = head->next;
259. int count = 0;
260. while (p != NULL) {
261. if (strcmp(p->data.className, className) == 0) {
262. count++;
263. }
264. p = p->next;
265. }
266. printf("班级 %s 共有 %d 名学生。\n", className, count);
267. }
268. void statisticsByGender(Node\* head) {
269. if (head->next == NULL) {
270. printf("当前无任何学生信息。\n");
271. return;
272. }
273. int male\_count = 0;
274. int female\_count = 0;
275. Node\* p = head->next;
276. while (p != NULL) {
277. if (strcmp(p->data.gender, "男") == 0) {
278. male\_count++;
279. }
280. else if (strcmp(p->data.gender, "女") == 0) {
281. female\_count++;
282. }
283. p = p->next;
284. }
285. printf("学生总数中，男生有 %d 人，女生有 %d 人。\n", male\_count, female\_count);
286. }
287. void statisticsMenu(Node\* head) {
288. int choice;
289. do {
290. printf("\n--- 统计记录模块 ---\n");
291. printf("  1. 按班级统计\n");
292. printf("  2. 按性别统计\n");
293. printf("  0. 返回主菜单\n");
294. printf("---------------------\n");
295. printf("请输入您的选择: ");
296. scanf("%d", &choice);
297. clearInputBuffer();
298. switch (choice) {
299. case 1: statisticsByClass(head); break;
300. case 2: statisticsByGender(head); break;
301. case 0: break;
302. default: printf("无效选择，请重新输入。\n");
303. }
304. } while (choice != 0);
305. }
306. *// 5. 输出记录模块*
307. void displayAllStudents(Node\* head) {
308. if (head->next == NULL) {
309. printf("当前无任何学生信息。\n");
310. return;
311. }
312. printf("\n======================= 所有学生信息 =======================\n");
313. printf("%-15s %-15s %-10s %-10s %-15s\n", "学号", "姓名", "性别", "年龄", "班级");
314. printf("------------------------------------------------------------\n");
315. Node\* p = head->next;
316. while (p != NULL) {
317. printf("%-15s %-15s %-10s %-10d %-15s\n",
318. p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);
319. p = p->next;
320. }
321. printf("============================================================\n");
322. }
323. *// 6. 文件操作模块*
324. void saveToFile(Node\* head, const char\* filename) {
325. FILE\* fp = fopen(filename, "w");
326. if (fp == NULL) {
327. printf("打开文件 %s 失败！\n", filename);
328. return;
329. }
330. Node\* p = head->next;
331. while (p != NULL) {
332. fprintf(fp, "%s %s %s %d %s\n",
333. p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);
334. p = p->next;
335. }
336. fclose(fp);
337. printf("数据已成功保存到 %s 文件中。\n", filename);
338. }
339. Node\* loadFromFile(const char\* filename) {
340. *// 创建一个带头节点的空链表*
341. Node\* head = (Node\*)malloc(sizeof(Node));
342. if (head == NULL) {
343. printf("内存分配失败！\n");
344. exit(1);
345. }
346. head->next = NULL;
347. FILE\* fp = fopen(filename, "r");
348. if (fp == NULL) {
349. printf("未找到数据文件 %s，将创建一个新系统。\n", filename);
350. return head;
351. }
352. Student temp;
353. while (fscanf(fp, "%s %s %s %d %s", temp.id, temp.name, temp.gender, &temp.age, temp.className) != EOF) {
354. Node\* newNode = (Node\*)malloc(sizeof(Node));
355. if (newNode == NULL) {
356. printf("内存分配失败！\n");
357. continue;
358. }
359. newNode->data = temp;
360. newNode->next = NULL;
361. *// 插入到链表尾部*
362. Node\* p = head;
363. while (p->next != NULL) {
364. p = p->next;
365. }
366. p->next = newNode;
367. }
368. fclose(fp);
369. printf("已从 %s 文件加载数据。\n", filename);
370. return head;
371. }
372. *// 辅助函数：释放链表*
373. void freeList(Node\* head) {
374. Node\* p = head;
375. Node\* temp;
376. while (p != NULL) {
377. temp = p;
378. p = p->next;
379. free(temp);
380. }
381. }
382. main.cpp (主函数文件)
383. #define \_CRT\_SECURE\_NO\_WARNINGS
384. #include "student.h"
385. #define FILENAME "student.txt"
386. void showMenu() {
387. printf("\n========= 学生信息管理系统 =========\n");
388. printf("||                                ||\n");
389. printf("||    1. 录入学生信息             ||\n");
390. printf("||    2. 查询学生信息             ||\n");
391. printf("||    3. 更新学生信息             ||\n");
392. printf("||    4. 统计学生信息             ||\n");
393. printf("||    5. 显示所有信息             ||\n");
394. printf("||                                ||\n");
395. printf("||    0. 退出系统 (Exit)          ||\n");
396. printf("||                                ||\n");
397. printf("====================================\n");
398. printf("请输入您的选择 [0-5]: ");
399. }
400. int main() {
401. *// 程序启动时，从文件加载数据*
402. Node\* head = loadFromFile(FILENAME);
403. int choice;
404. do {
405. showMenu();
406. scanf("%d", &choice);
407. clearInputBuffer(); *// 吸收多余的换行符*
408. switch (choice) {
409. case 1:
410. addStudent(head);
411. saveToFile(head, FILENAME);
412. break;
413. case 2:
414. queryMenu(head);
415. break;
416. case 3:
417. updateMenu(head);
418. saveToFile(head, FILENAME);
419. break;
420. case 4:
421. statisticsMenu(head);
422. break;
423. case 5:
424. displayAllStudents(head);
425. break;
426. case 0:
427. {   FILE\* fp = fopen(FILENAME, "w"); }*// 以写入模式打开文件，会直接清空内容*
428. printf("正在退出系统...感谢使用！\n");
429. break;
430. default:
431. printf("无效的输入，请输入 0-5 之间的数字。\n");
432. break;
433. }
434. printf("\n按 Enter 键继续...");
435. getchar(); *// 暂停程序，等待用户按键*
436. system("cls"); *// 清屏*
437. } while (choice != 0);
438. *// 退出程序前，释放链表内存*
439. freeList(head);
440. return 0;
441. }