

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 | | | | | | | | |

## 2．算法设计

### 1. 设计思路：

本系统采用 "单链表数据结构 + 文件持久化" 的设计思路，核心在于通过单链表实现内存中的数据管理，同时利用文件操作实现数据的持久化存储。系统启动时自动从文件加载数据到链表，用户操作时直接修改链表数据，操作完成后保存回文件，确保数据不丢失。

### 2. 主要步骤：

1. **数据结构定义**：在头文件中定义Student结构体存储学生信息，Node结构体构成单链表节点。
2. **链表操作实现**：实现链表的创建、插入、删除、遍历等基础操作。
3. **文件操作实现**：
   * saveToFile函数：将链表数据按格式写入文件
   * loadFromFile函数：从文件读取数据重建链表
4. **功能模块开发**：基于链表操作实现输入、查询、更新、统计、输出五大功能模块
5. **主菜单集成**：设计交互式菜单，关联各功能模块
6. **数据持久化**：在数据变更后及时保存到文件，确保数据不丢失

### 3. 功能模块算法设计

#### 4. 输入记录模块 (addStudent)

* **目标**：向单链表中添加新的学生记录
* **算法**：
  1. 动态分配新节点内存
  2. 输入学生信息并进行学号唯一性检查
  3. 若学号唯一，使用尾插法将节点插入链表
  4. 释放无效节点内存，避免内存泄漏

#### 5. 更新记录模块 (modifyStudent, deleteStudent)

* **修改算法**：
  1. 输入目标学号，遍历链表查找节点
  2. 找到后更新节点中的学生信息
  3. 未找到时提示用户
* **删除算法**：
  1. 输入目标学号，定义前驱指针和当前指针
  2. 同时移动双指针查找目标节点
  3. 找到后修改前驱指针指向，断开目标节点
  4. 释放目标节点内存，更新链表计数

#### 6. 查询与统计模块

* **查询算法**：
  1. 支持按学号精确查询和按姓名模糊查询
  2. 遍历链表对比查询条件
  3. 匹配成功时显示学生信息
* **统计算法**：
  1. 初始化计数器，遍历链表
  2. 按班级或性别条件统计符合要求的节点数量
  3. 输出统计结果

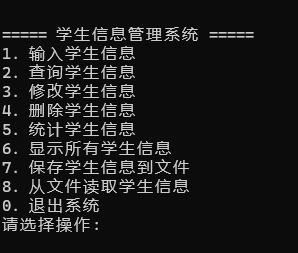
#### 7. 文件持久化模块 (saveToFile, loadFromFile)

* **保存算法**：
  1. 以写入模式打开文件，清空原有内容
  2. 遍历链表，按格式将每个节点数据写入文件
  3. 关闭文件并提示保存成功
* **加载算法**：
  1. 以读取模式打开文件，处理文件不存在情况
  2. 逐行读取文件数据，创建新节点并插入链表
  3. 构建完整链表后关闭文件

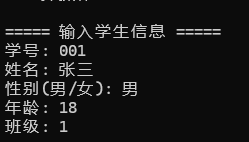
## 3．算法实现

### 1. 程序主页面，启动与加载：

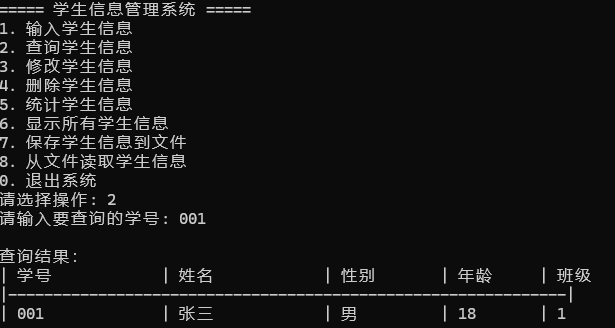
系统启动后自动从student.txt文件加载数据，显示主菜单等待用户操作：



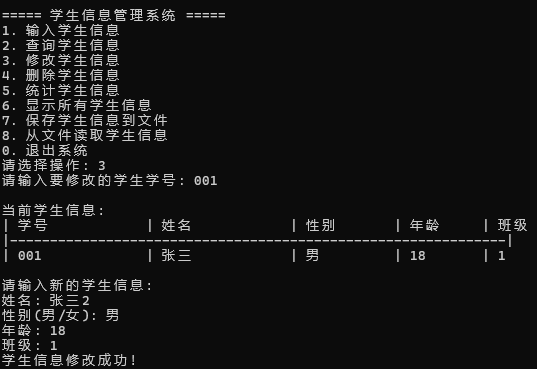
### 2. 录入学生信息：



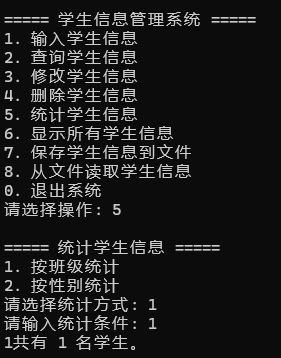
### 3. 查询学生信息：



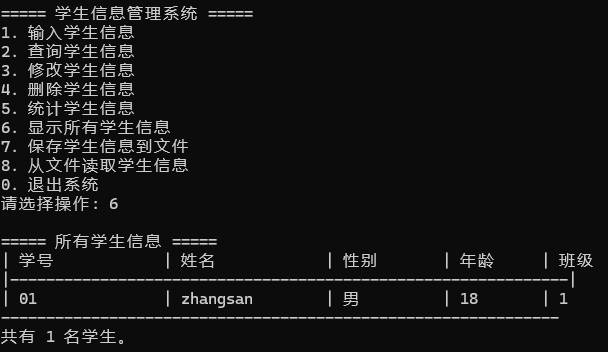
### 4. 更新学生信息：



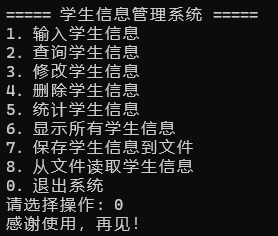
### 5. 统计学生信息：



### 6. 显示所有信息：



### 7. 退出系统：



## 四．心得体会

通过本次 C 语言课程设计，我深入理解了单链表数据结构的应用，成功实现了一个完整的学生信息管理系统。在实现过程中，我收获了以下体会：

首先，单链表的指针操作是核心难点。从节点的动态分配到指针的正确指向，每一步都需要严谨的逻辑。例如，在删除节点时，必须同时处理前驱指针和内存释放，否则会导致链表断裂或内存泄漏。这种 "指针思维" 的培养为后续学习更复杂的数据结构奠定了基础。

其次，文件持久化功能的实现让我体会到数据存储的重要性。在saveToFile和loadFromFile函数中，数据格式的一致性是关键。任何字段的顺序或类型不匹配都会导致数据读取错误，这让我明白接口设计的严谨性至关重要。同时，处理文件不存在等边界情况的经验，提升了我编写健壮程序的能力。

另外，模块化设计思想让系统开发更加清晰。将功能拆分为输入、查询等独立模块后，每个模块的职责单一，不仅降低了编码难度，也便于调试和维护。例如，查询模块可以单独测试遍历和匹配逻辑，而不影响其他功能。

在遇到的问题中，学号唯一性校验和中文显示问题印象最为深刻。通过遍历链表对比学号实现了唯一性检查，而通过设置文件编码为 GB2312 解决了中文乱码问题。这些经历让我学会了如何通过逻辑分析和调试工具解决实际问题。

此次设计也让我认识到单链表的优缺点：插入删除操作灵活，但查询效率较低。在后续优化中，可以考虑对链表进行排序并结合二分查找，提升查询效率。同时，增加数据校验功能（如年龄范围检查）和图形化界面，能让系统更接近实际应用。

总体而言，这次课程设计不仅巩固了 C 语言知识，更让我体验了从需求分析到代码实现的完整软件开发流程，提升了问题解决能力和工程化思维。

## 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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: 2022.12: 108.

## 源代码

### 1. student.h (头文件)

|  |
| --- |
| #ifndef STUDENT\_H  #define STUDENT\_H  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  // 学生信息结构体  typedef struct Student {  char id[20];  char name[50];  char gender[10];  int age;  char className[50];  } Student;  // 链表节点结构体  typedef struct Node {  Student data;  struct Node\* next;  } Node;  // 函数原型声明  void addStudent(Node\* head);  void queryMenu(Node\* head);  void updateMenu(Node\* head);  void statisticsMenu(Node\* head);  void displayAllStudents(Node\* head);  void saveToFile(Node\* head, const char\* filename);  Node\* loadFromFile(const char\* filename);  void clearInputBuffer();  void freeList(Node\* head);  #endif // STUDENT\_H |

### 2. student.cpp (函数实现文件)

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include "student.h"  // 清空输入缓冲区  void clearInputBuffer() {  int c;  while ((c = getchar()) != '\n' && c != EOF);  }  // 检查学号是否唯一  int isIdUnique(Node\* head, const char\* id) {  Node\* p = head->next;  while (p != NULL) {  if (strcmp(p->data.id, id) == 0) {  return 0; // 不唯一  }  p = p->next;  }  return 1; // 唯一  }  // 输入记录模块  void addStudent(Node\* head) {  Node\* newNode = (Node\*)malloc(sizeof(Node));  if (newNode == NULL) {  printf("内存分配失败！\n");  return;  }  printf("请输入学号: ");  scanf("%s", newNode->data.id);  clearInputBuffer();  if (!isIdUnique(head, newNode->data.id)) {  printf("错误：该学号已存在！\n");  free(newNode);  return;  }  printf("请输入姓名: ");  scanf("%s", newNode->data.name);  clearInputBuffer();  printf("请输入性别: ");  scanf("%s", newNode->data.gender);  clearInputBuffer();  printf("请输入年龄: ");  scanf("%d", &newNode->data.age);  clearInputBuffer();  printf("请输入班级: ");  scanf("%s", newNode->data.className);  clearInputBuffer();  newNode->next = NULL;  // 找到链表尾部并插入  Node\* p = head;  while (p->next != NULL) {  p = p->next;  }  p->next = newNode;  printf("\n学生信息添加成功！\n");  }  // 查询记录模块  void searchStudentById(Node\* head) {  if (head->next == NULL) {  printf("当前无任何学生信息。\n");  return;  }  char id[20];  printf("请输入要查询的学号: ");  scanf("%s", id);  clearInputBuffer();  Node\* p = head->next;  int found = 0;  while (p != NULL) {  if (strcmp(p->data.id, id) == 0) {  printf("\n查询成功，学生信息如下:\n");  printf("学号: %s, 姓名: %s, 性别: %s, 年龄: %d, 班级: %s\n",  p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);  found = 1;  break;  }  p = p->next;  }  if (!found) {  printf("未找到学号为 %s 的学生。\n", id);  }  }  void searchStudentByName(Node\* head) {  if (head->next == NULL) {  printf("当前无任何学生信息。\n");  return;  }  char name[50];  printf("请输入要查询的姓名: ");  scanf("%s", name);  clearInputBuffer();  Node\* p = head->next;  int found = 0;  printf("\n查询结果如下:\n");  while (p != NULL) {  if (strcmp(p->data.name, name) == 0) {  printf("学号: %s, 姓名: %s, 性别: %s, 年龄: %d, 班级: %s\n",  p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);  found++;  }  p = p->next;  }  if (found == 0) {  printf("未找到姓名为 %s 的学生。\n", name);  } else {  printf("共找到 %d 条记录。\n", found);  }  }  void queryMenu(Node\* head) {  int choice;  do {  printf("\n--- 查询记录模块 ---\n");  printf(" 1. 按学号查询\n");  printf(" 2. 按姓名查询\n");  printf(" 0. 返回主菜单\n");  printf("---------------------\n");  printf("请输入您的选择: ");  scanf("%d", &choice);  clearInputBuffer();  switch (choice) {  case 1: searchStudentById(head); break;  case 2: searchStudentByName(head); break;  case 0: break;  default: printf("无效选择，请重新输入。\n");  }  } while (choice != 0);  }  // 更新记录模块  void modifyStudent(Node\* head) {  if (head->next == NULL) {  printf("当前无任何学生信息。\n");  return;  }  char id[20];  printf("请输入要修改的学生的学号: ");  scanf("%s", id);  clearInputBuffer();  Node\* p = head->next;  int found = 0;  while (p != NULL) {  if (strcmp(p->data.id, id) == 0) {  printf("已找到该学生，请输入新信息:\n");  printf("请输入新姓名: ");  scanf("%s", p->data.name);  clearInputBuffer();  printf("请输入新性别: ");  scanf("%s", p->data.gender);  clearInputBuffer();  printf("请输入新年龄: ");  scanf("%d", &p->data.age);  clearInputBuffer();  printf("请输入新班级: ");  scanf("%s", p->data.className);  clearInputBuffer();  printf("\n学生信息修改成功！\n");  found = 1;  break;  }  p = p->next;  }  if (!found) {  printf("未找到学号为 %s 的学生。\n", id);  }  }  void deleteStudent(Node\* head) {  if (head->next == NULL) {  printf("当前无任何学生信息。\n");  return;  }  char id[20];  printf("请输入要删除的学生的学号: ");  scanf("%s", id);  clearInputBuffer();  Node\* prev = head;  Node\* curr = head->next;  int found = 0;  while (curr != NULL) {  if (strcmp(curr->data.id, id) == 0) {  prev->next = curr->next;  free(curr);  printf("学号为 %s 的学生信息已删除。\n", id);  found = 1;  break;  }  prev = curr;  curr = curr->next;  }  if (!found) {  printf("未找到学号为 %s 的学生。\n", id);  }  }  void updateMenu(Node\* head) {  int choice;  do {  printf("\n--- 更新记录模块 ---\n");  printf(" 1. 修改学生信息\n");  printf(" 2. 删除学生信息\n");  printf(" 0. 返回主菜单\n");  printf("---------------------\n");  printf("请输入您的选择: ");  scanf("%d", &choice);  clearInputBuffer();  switch (choice) {  case 1: modifyStudent(head); break;  case 2: deleteStudent(head); break;  case 0: break;  default: printf("无效选择，请重新输入。\n");  }  } while (choice != 0);  }  // 统计记录模块  void statisticsByClass(Node\* head) {  if (head->next == NULL) {  printf("当前无任何学生信息。\n");  return;  }  char className[50];  printf("请输入要统计的班级名称: ");  scanf("%s", className);  clearInputBuffer();  Node\* p = head->next;  int count = 0;  while (p != NULL) {  if (strcmp(p->data.className, className) == 0) {  count++;  }  p = p->next;  }  printf("班级 %s 共有 %d 名学生。\n", className, count);  }  void statisticsByGender(Node\* head) {  if (head->next == NULL) {  printf("当前无任何学生信息。\n");  return;  }  int male\_count = 0;  int female\_count = 0;  Node\* p = head->next;  while (p != NULL) {  if (strcmp(p->data.gender, "男") == 0) {  male\_count++;  } else if (strcmp(p->data.gender, "女") == 0) {  female\_count++;  }  p = p->next;  }  printf("学生总数中，男生有 %d 人，女生有 %d 人。\n", male\_count, female\_count);  }  void statisticsMenu(Node\* head) {  int choice;  do {  printf("\n--- 统计记录模块 ---\n");  printf(" 1. 按班级统计\n");  printf(" 2. 按性别统计\n");  printf(" 0. 返回主菜单\n");  printf("---------------------\n");  printf("请输入您的选择: ");  scanf("%d", &choice);  clearInputBuffer();  switch (choice) {  case 1: statisticsByClass(head); break;  case 2: statisticsByGender(head); break;  case 0: break;  default: printf("无效选择，请重新输入。\n");  }  } while (choice != 0);  }  // 输出记录模块  void displayAllStudents(Node\* head) {  if (head->next == NULL) {  printf("当前无任何学生信息。\n");  return;  }  printf("\n======================= 所有学生信息 =======================\n");  printf("%-15s %-15s %-10s %-10s %-15s\n", "学号", "姓名", "性别", "年龄", "班级");  printf("------------------------------------------------------------\n");  Node\* p = head->next;  while (p != NULL) {  printf("%-15s %-15s %-10s %-10d %-15s\n",  p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);  p = p->next;  }  printf("============================================================\n");  }  // 文件操作模块  void saveToFile(Node\* head, const char\* filename) {  FILE\* fp = fopen(filename, "w");  if (fp == NULL) {  printf("打开文件 %s 失败！\n", filename);  return;  }  Node\* p = head->next;  while (p != NULL) {  fprintf(fp, "%s %s %s %d %s\n",  p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);  p = p->next;  }  fclose(fp);  printf("数据已成功保存到 %s 文件中。\n", filename);  }  Node\* loadFromFile(const char\* filename) {  // 创建一个带头节点的空链表  Node\* head = (Node\*)malloc(sizeof(Node));  if (head == NULL) {  printf("内存分配失败！\n");  exit(1);  }  head->next = NULL;  FILE\* fp = fopen(filename, "r");  if (fp == NULL) {  printf("未找到数据文件 %s，将创建一个新系统。\n", filename);  return head;  }  Student temp;  while (fscanf(fp, "%s %s %s %d %s", temp.id, temp.name, temp.gender, &temp.age, temp.className) != EOF) {  Node\* newNode = (Node\*)malloc(sizeof(Node));  if (newNode == NULL) {  printf("内存分配失败！\n");  continue;  }  newNode->data = temp;  newNode->next = NULL;  // 插入到链表尾部  Node\* p = head;  while (p->next != NULL) {  p = p->next;  }  p->next = newNode;  }  fclose(fp);  printf("已从 %s 文件加载数据。\n", filename);  return head;  }  // 辅助函数：释放链表  void freeList(Node\* head) {  Node\* p = head;  Node\* temp;  while (p != NULL) {  temp = p;  p = p->next;  free(temp);  }  } |

### 3. main.cpp (主函数文件)

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include "student.h"  #define FILENAME "student.txt"  void showMenu() {  printf("\n========= 学生信息管理系统 =========\n");  printf("|| ||\n");  printf("|| 1. 录入学生信息 ||\n");  printf("|| 2. 查询学生信息 ||\n");  printf("|| 3. 更新学生信息 ||\n");  printf("|| 4. 统计学生信息 ||\n");  printf("|| 5. 显示所有信息 ||\n");  printf("|| ||\n");  printf("|| 0. 退出系统 (Exit) ||\n");  printf("|| ||\n");  printf("====================================\n");  printf("请输入您的选择 [0-5]: ");  }  int main() {  // 程序启动时，从文件加载数据  Node\* head = loadFromFile(FILENAME);  int choice;  do {  showMenu();  scanf("%d", &choice);  clearInputBuffer(); // 吸收多余的换行符  switch (choice) {  case 1:  addStudent(head);  saveToFile(head, FILENAME);  break;  case 2:  queryMenu(head);  break;  case 3:  updateMenu(head);  saveToFile(head, FILENAME);  break;  case 4:  statisticsMenu(head);  break;  case 5:  displayAllStudents(head);  break;  case 0:  printf("正在退出系统...感谢使用！\n");  break;  default:  printf("无效的输入，请输入 0-5 之间的数字。\n");  break;  }  printf("\n按 Enter 键继续...");  getchar(); // 暂停程序，等待用户按键  system("cls"); // 清屏  } while (choice != 0);  // 退出程序前，释放链表内存  freeList(head);  return 0;  } |