

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

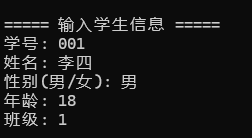
本系统采用**动态数组**作为核心数据结构，遵循 “数据连续存储 + 容量动态扩展” 原则。与单链表方案相比，动态数组通过下标直接访问数据，提升查询效率，更适合以查询为主的场景。系统通过realloc实现内存动态管理，当数据量超过当前容量时自动扩容，确保存储效率与灵活性的平衡。

### 2. 主要步骤

1. **数据结构定义**：定义Student结构体存储学生信息，StudentArray结构体管理动态数组，包含数据指针、当前容量和元素计数。
2. **动态扩容实现**：输入数据时检测容量，不足时通过realloc扩展为 2 倍容量，避免频繁内存操作。
3. **模块化功能开发**：基于动态数组实现输入、查询、更新、统计、输出模块，利用数组下标直接访问特性优化查询操作。
4. **文件持久化**：二进制格式读写数组数据，保存时写入容量和计数，读取时按格式重建数组。

### 3. 功能模块算法设计

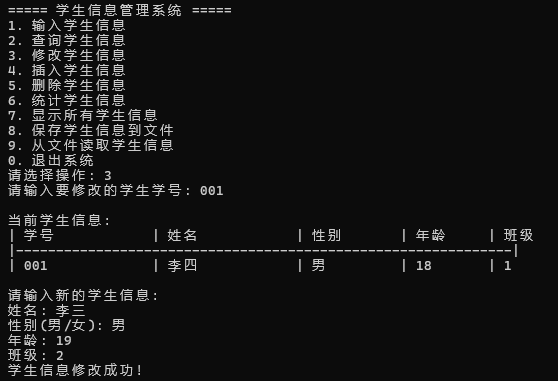
#### 4. 输入记录模块（addStudent）

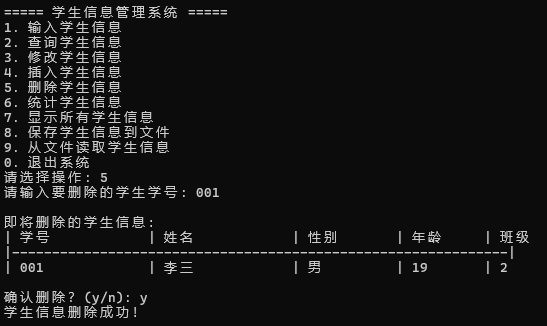


* **算法流程**：
  1. 检查当前数组容量，若count >= capacity，则使用realloc扩展为 2 倍容量。
  2. 接收学生信息并校验学号唯一性（遍历数组对比）。
  3. 将新数据存入数组末尾，count自增。
* **核心代码示例**：

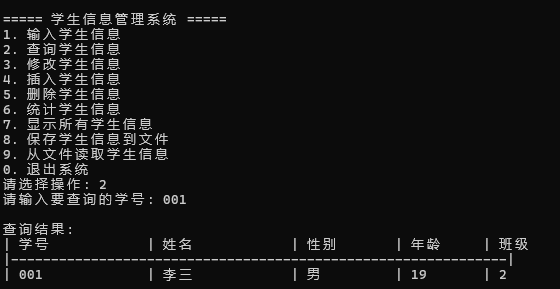
|  |
| --- |
| if (array->count >= array->capacity) {  int newCap = array->capacity == 0 ? 10 : array->capacity \* 2;  Student\* newData = (Student\*)realloc(array->data, newCap \* sizeof(Student));  // 错误处理及指针更新  } |

#### 更新记录模块（modifyStudent/deleteStudent）

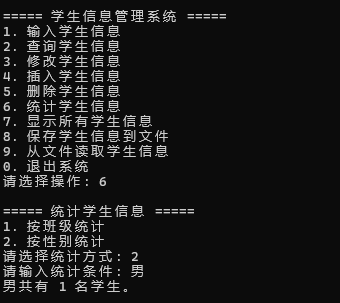


* **修改算法**：
  1. 输入学号，遍历数组找到对应下标。
  2. 直接通过下标更新该位置的学生信息。
* **删除算法**：
  1. 找到目标下标i，将i之后的元素依次前移（data[i] = data[i+1]）。
  2. count自减，若count < capacity/4则收缩容量（可选）。
     

#### 查询与统计模块



* **查询算法**：
  + 按学号查询：遍历数组对比id，时间复杂度 O (n)；
  + 优化方向：排序后使用二分查找，复杂度降为 O (log n)。



* **统计算法**：

遍历数组计数，按班级或性别匹配条件，时间复杂度 O (n)。

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

* **保存算法**：
  1. 以二进制模式打开文件，先写入capacity和count。
  2. 用fwrite一次性写入数组数据。
* **加载算法**：
  1. 读取容量和计数，分配对应内存。
  2. 用fread读取数据到数组，重建动态数组。

## 3．算法实现（关键代码示例）

### 1. 动态数组核心结构

|  |
| --- |
| typedef struct {  Student\* data; // 学生数据指针  int capacity; // 当前容量  int count; // 元素数量  } StudentArray; |

### 2. 输入与扩容实现

|  |
| --- |
| void addStudent(StudentArray\* array) {  // 动态扩容  if (array->count >= array->capacity) {  int newCap = array->capacity == 0 ? 10 : array->capacity \* 2;  Student\* newData = (Student\*)realloc(array->data, newCap \* sizeof(Student));  if (!newData) { printf("内存分配失败！"); exit(1); }  array->data = newData;  array->capacity = newCap;  }    // 输入数据（略）  array->data[array->count] = newStu;  array->count++;  } |

### 3. 删除操作实现

|  |
| --- |
| void deleteStudent(StudentArray\* array, char\* id) {  int index = findIndexById(array, id);  if (index == -1) return;    // 元素前移  for (int i = index; i < array->count - 1; i++) {  array->data[i] = array->data[i + 1];  }  array->count--;    // 收缩容量（可选）  if (array->count < array->capacity / 4 && array->capacity > 10) {  int newCap = array->capacity / 2;  Student\* newData = (Student\*)realloc(array->data, newCap \* sizeof(Student));  if (newData) {  array->data = newData;  array->capacity = newCap;  }  }  } |

### 4. 文件读写实现

|  |
| --- |
| void saveToFile(StudentArray\* array, const char\* filename) {  FILE\* fp = fopen(filename, "wb");  if (!fp) { printf("文件打开失败！"); return; }    // 写入容量和计数  fwrite(&array->capacity, sizeof(int), 1, fp);  fwrite(&array->count, sizeof(int), 1, fp);  // 写入数据  fwrite(array->data, sizeof(Student), array->count, fp);    fclose(fp);  }  StudentArray\* loadFromFile(const char\* filename) {  StudentArray\* array = (StudentArray\*)malloc(sizeof(StudentArray));  array->data = NULL;  array->capacity = 0;  array->count = 0;    FILE\* fp = fopen(filename, "rb");  if (!fp) { return array; }    // 读取容量和计数  fread(&array->capacity, sizeof(int), 1, fp);  fread(&array->count, sizeof(int), 1, fp);    // 分配内存并读取数据  array->data = (Student\*)malloc(array->capacity \* sizeof(Student));  fread(array->data, sizeof(Student), array->count, fp);    fclose(fp);  return array;  } |

## 4．心得体会

### 1. 动态数组与单链表对比

* **效率优势**：动态数组通过下标访问数据，查询时间复杂度为 O (1)，比单链表的 O (n) 更高效；
* **操作代价**：插入 / 删除需移动元素，平均复杂度 O (n)，不如单链表的指针操作灵活；
* **内存管理**：动态数组通过realloc减少内存碎片，但扩容时可能产生数据拷贝开销。

### 2. 关键技术收获

* **动态扩容策略**：采用 2 倍扩容因子，平衡内存效率与操作开销；
* **数据持久化**：二进制文件读写比文本格式更高效，需注意数据对齐问题；
* **边界处理**：处理空数组、容量收缩等场景，提升程序健壮性。

### 3. 优化方向

* **查询优化**：对数组排序后使用二分查找，提升大规模数据查询效率；
* **内存优化**：采用更精细的扩容策略（如 1.5 倍扩容），减少内存浪费；
* **数据校验**：增加年龄范围、班级格式等输入校验，提升数据合法性。

## 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.

## 完整代码（动态数组版）

### 1. 完整代码实现

#### 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 {  char id[20]; // 学号（唯一标识）  char name[50]; // 姓名  char gender[10]; // 性别  int age; // 年龄  char className[50]; // 班级  } Student;  // 动态数组管理结构体  typedef struct {  Student\* data; // 数据指针  int capacity; // 当前容量  int count; // 元素数量  } StudentArray;  // 函数声明  void initArray(StudentArray\* array);  void addStudent(StudentArray\* array);  void deleteStudent(StudentArray\* array, char\* id);  void modifyStudent(StudentArray\* array, char\* id);  void queryStudentById(StudentArray\* array, char\* id);  void queryStudentByName(StudentArray\* array, char\* name);  void statisticsByClass(StudentArray\* array, char\* className);  void statisticsByGender(StudentArray\* array);  void displayAllStudents(StudentArray\* array);  void saveToFile(StudentArray\* array, const char\* filename);  StudentArray\* loadFromFile(const char\* filename);  void freeArray(StudentArray\* array);  int findIndexById(StudentArray\* array, char\* id);  void clearInputBuffer();  #endif // STUDENT\_H |

#### student.c 功能实现

|  |
| --- |
| #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);  }  // 初始化动态数组  void initArray(StudentArray\* array) {  array->data = NULL;  array->capacity = 0;  array->count = 0;  }  // 动态扩容核心函数  void ensureCapacity(StudentArray\* array) {  if (array->count >= array->capacity) {  int newCapacity = array->capacity == 0 ? 10 : array->capacity \* 2;  Student\* newData = (Student\*)realloc(array->data, newCapacity \* sizeof(Student));  if (newData == NULL) {  printf("内存分配失败！程序终止\n");  exit(1);  }  array->data = newData;  array->capacity = newCapacity;  }  }  // 输入学生信息  void addStudent(StudentArray\* array) {  ensureCapacity(array);  Student newStu;    printf("请输入学号: ");  scanf("%s", newStu.id);  clearInputBuffer();    // 校验学号唯一性  if (findIndexById(array, newStu.id) != -1) {  printf("错误：该学号已存在！\n");  return;  }    printf("请输入姓名: "); scanf("%s", newStu.name);  printf("请输入性别: "); scanf("%s", newStu.gender);  printf("请输入年龄: "); scanf("%d", &newStu.age);  printf("请输入班级: "); scanf("%s", newStu.className);  clearInputBuffer();    array->data[array->count] = newStu;  array->count++;  printf("学生信息添加成功！\n");  }  // 按学号查找下标  int findIndexById(StudentArray\* array, char\* id) {  for (int i = 0; i < array->count; i++) {  if (strcmp(array->data[i].id, id) == 0) {  return i;  }  }  return -1;  }  // 按学号查询学生  void queryStudentById(StudentArray\* array, char\* id) {  int index = findIndexById(array, id);  if (index == -1) {  printf("未找到学号为 %s 的学生！\n", id);  return;  }    printf("\n查询结果：\n");  printf("| %-15s | %-15s | %-10s | %-8s | %-20s |\n",  "学号", "姓名", "性别", "年龄", "班级");  printf("|--------------------------------------------------------------|\n");  printf("| %-15s | %-15s | %-10s | %-8d | %-20s |\n",  array->data[index].id, array->data[index].name,  array->data[index].gender, array->data[index].age,  array->data[index].className);  }  // 按姓名查询学生  void queryStudentByName(StudentArray\* array, char\* name) {  int found = 0;  printf("\n查询结果：\n");  printf("| %-15s | %-15s | %-10s | %-8s | %-20s |\n",  "学号", "姓名", "性别", "年龄", "班级");  printf("|--------------------------------------------------------------|\n");    for (int i = 0; i < array->count; i++) {  if (strcmp(array->data[i].name, name) == 0) {  printf("| %-15s | %-15s | %-10s | %-8d | %-20s |\n",  array->data[i].id, array->data[i].name,  array->data[i].gender, array->data[i].age,  array->data[i].className);  found++;  }  }    if (found == 0) {  printf("未找到姓名为 %s 的学生！\n", name);  } else {  printf("--------------------------------------------------------------\n");  printf("共找到 %d 条记录。\n", found);  }  }  // 修改学生信息  void modifyStudent(StudentArray\* array, char\* id) {  int index = findIndexById(array, id);  if (index == -1) {  printf("未找到学号为 %s 的学生！\n", id);  return;  }    printf("请输入新姓名: "); scanf("%s", array->data[index].name);  printf("请输入新性别: "); scanf("%s", array->data[index].gender);  printf("请输入新年龄: "); scanf("%d", &array->data[index].age);  printf("请输入新班级: "); scanf("%s", array->data[index].className);  clearInputBuffer();    printf("学生信息修改成功！\n");  }  // 删除学生信息  void deleteStudent(StudentArray\* array, char\* id) {  int index = findIndexById(array, id);  if (index == -1) {  printf("未找到学号为 %s 的学生！\n", id);  return;  }    // 元素前移  for (int i = index; i < array->count - 1; i++) {  array->data[i] = array->data[i + 1];  }  array->count--;    // 收缩容量（当元素数量少于容量1/4时）  if (array->count < array->capacity / 4 && array->capacity > 10) {  int newCapacity = array->capacity / 2;  Student\* newData = (Student\*)realloc(array->data, newCapacity \* sizeof(Student));  if (newData != NULL) {  array->data = newData;  array->capacity = newCapacity;  }  }    printf("学号为 %s 的学生信息已删除！\n", id);  }  // 按班级统计  void statisticsByClass(StudentArray\* array, char\* className) {  int count = 0;  for (int i = 0; i < array->count; i++) {  if (strcmp(array->data[i].className, className) == 0) {  count++;  }  }  printf("班级 %s 共有 %d 名学生。\n", className, count);  }  // 按性别统计  void statisticsByGender(StudentArray\* array) {  int male = 0, female = 0;  for (int i = 0; i < array->count; i++) {  if (strcmp(array->data[i].gender, "男") == 0) {  male++;  } else if (strcmp(array->data[i].gender, "女") == 0) {  female++;  }  }  printf("学生总数中，男生有 %d 人，女生有 %d 人。\n", male, female);  }  // 显示所有学生  void displayAllStudents(StudentArray\* array) {  if (array->count == 0) {  printf("暂无学生信息！\n");  return;  }    printf("\n======================= 所有学生信息 =======================\n");  printf("| %-15s | %-15s | %-10s | %-8s | %-20s |\n",  "学号", "姓名", "性别", "年龄", "班级");  printf("|--------------------------------------------------------------|\n");    for (int i = 0; i < array->count; i++) {  printf("| %-15s | %-15s | %-10s | %-8d | %-20s |\n",  array->data[i].id, array->data[i].name,  array->data[i].gender, array->data[i].age,  array->data[i].className);  }    printf("--------------------------------------------------------------\n");  printf("共有 %d 名学生。\n", array->count);  }  // 保存到文件（二进制格式）  void saveToFile(StudentArray\* array, const char\* filename) {  FILE\* fp = fopen(filename, "wb");  if (fp == NULL) {  printf("文件打开失败！\n");  return;  }    // 写入容量、计数及数据  fwrite(&array->capacity, sizeof(int), 1, fp);  fwrite(&array->count, sizeof(int), 1, fp);  fwrite(array->data, sizeof(Student), array->count, fp);    fclose(fp);  printf("数据已成功保存到 %s 文件！\n", filename);  }  // 从文件加载  StudentArray\* loadFromFile(const char\* filename) {  StudentArray\* array = (StudentArray\*)malloc(sizeof(StudentArray));  initArray(array);    FILE\* fp = fopen(filename, "rb");  if (fp == NULL) {  printf("未找到数据文件，将创建新系统。\n");  return array;  }    // 读取容量和计数  fread(&array->capacity, sizeof(int), 1, fp);  fread(&array->count, sizeof(int), 1, fp);    // 分配内存并读取数据  array->data = (Student\*)malloc(array->capacity \* sizeof(Student));  fread(array->data, sizeof(Student), array->count, fp);    fclose(fp);  printf("已从 %s 加载 %d 条学生信息！\n", filename, array->count);  return array;  }  // 释放数组内存  void freeArray(StudentArray\* array) {  free(array->data);  free(array);  } |

#### main.c 主函数

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include "student.h"  #define FILENAME "students.dat" // 数据文件名  // 显示主菜单  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. 按学号查询 ||\n");  printf("|| 3. 按姓名查询 ||\n");  printf("|| 4. 修改学生信息 ||\n");  printf("|| 5. 删除学生信息 ||\n");  printf("|| 6. 按班级统计 ||\n");  printf("|| 7. 按性别统计 ||\n");  printf("|| 8. 显示所有学生 ||\n");  printf("|| 9. 保存数据到文件 ||\n");  printf("|| 0. 退出系统 ||\n");  printf("|| ||\n");  printf("====================================\n");  printf("请输入您的选择 [0-9]: ");  }  int main() {  // 加载数据  StudentArray\* array = loadFromFile(FILENAME);  int choice;  char id[20], name[50], className[50];    do {  showMenu();  scanf("%d", &choice);  clearInputBuffer();    switch (choice) {  case 1: addStudent(array); break;  case 2:  printf("请输入学号: ");  scanf("%s", id);  queryStudentById(array, id);  break;  case 3:  printf("请输入姓名: ");  scanf("%s", name);  queryStudentByName(array, name);  break;  case 4:  printf("请输入要修改的学号: ");  scanf("%s", id);  modifyStudent(array, id);  break;  case 5:  printf("请输入要删除的学号: ");  scanf("%s", id);  deleteStudent(array, id);  break;  case 6:  printf("请输入班级名称: ");  scanf("%s", className);  statisticsByClass(array, className);  break;  case 7: statisticsByGender(array); break;  case 8: displayAllStudents(array); break;  case 9: saveToFile(array, FILENAME); break;  case 0: printf("正在退出系统，感谢使用！\n"); break;  default: printf("无效选择，请重新输入！\n");  }    if (choice != 0) {  printf("\n按 Enter 键继续...");  getchar();  }  } while (choice != 0);  // 释放内存  freeArray(array);  return 0;  } |