班级序号\_\_\_\_\_

**华南理工大学广州学院**

**2019-2020学年度第2学期**

**课程名称： 数据结构大作业**

**题　　目： 学委作业管理系统**

**专业班级： 信息与计算科学**

**年　　级： 2018级**

**姓　　名： 谢继发**

**学　　号：** 201810096055

|  |  |  |  |
| --- | --- | --- | --- |
| 一级指标 | 二级指标 | 分值 | 评分及成绩 |
| 得分 |
| 作品完成度  （占50%） | 1.工作量达标，功能点完整，独立完成 | 50 |  |
| 2.设计合理性，逻辑条理性 |
| 3.功能测试 |
| 4.数据处理水平 |
| 5.特色鲜明，有设计亮点 |
| 文档撰写（30%） | 1.文字描述规范，语言通顺，结构完整 | 30 |  |
| 2.思路描述清晰，功能测试结果完整 |
| 3.结果分析合理 |
| 答辩成绩 （占20%） | 1.能够对设计进行合理说明 | 20 |  |
| 2.能较好的回答答辩所提问题，解释合理清晰 |
| 合计（百分制） | | |  |
| 总 评 | □优秀 □良好 □中等 □及格 □不及格 | | 签名 |

# 目录

[一、题目 1](#_Toc534375176)

[二、内容 1](#_Toc534375177)

[1.概述 1](#_Toc534375178)

[2.程序概要设计 1](#_Toc534375179)

[3.程序详细设计 2](#_Toc534375180)

[4.程序测试说明 6](#_Toc534375181)

三、[总结 8](#_Toc534375182)

[四、参考文献 9](#_Toc534375183)

1. **题目**
2. 学委作业管理系统
3. 选题理由

我选择这个题目的理由：第一、思路明确，目的性强，题目要求设置三个角色分别实现三个角色的增、删、查、改、排序。第二、该类型系统较为常见，容易实现。

1. 开发环境：VS2013

# 内容

## 概述

我对题目的理解是：设计三个角色，分别实现管理员管理作业、班级、学委任命等功能；实现学委管理作业、提交等功能；以及实现学生查看作业情况和搜索作业提交情况。跟其他管理系统实现相类似。

## 2.程序概要设计

该程序主要分为3个流程：

登录学委作业管理系统

选择角色进行相关操作

存储相关信息

## 3.程序详细设计

从上面的效果图回想自己的做题历程，有几个难题一直阻碍开发进程，也是我自认为的关键部分，现在对其进行详细分析：

1. 如何才不会头重脚轻？如何减少代码的重复率？怎样设计才能尽可能减少代码量？于是向着这个方向思考，发现学习java中只要同一包中的类可以随意调用。因此我只需要把我需要的方法通过#include来导入即可。于是分别写了homework.h、sumission.h、student.h、class.h用来分别存放各自的函数及结构体，提高代码的安全性及可读性，若要查看函数如何实现，则需要通过逐层查找来查看源码。
2. 如何实现学委的按时间范围查询作业、按时间升或降序进行排列？下面我详细说明一下我的实现思路：传入作业链表、开始时间和结束时间参数，若传入的开始时间比原有设置的作业开始时间小并且传入的结束时间比原有设置的结束时间大，则证明作业存在与这一区间中可以查询相应的作业，同时标记已找到的作业。附部分关键源码：

int searchAndPrintHomework2(homework \*h, char \* start\_date, char \* end\_date)

{

printf("Homework Information in date from %s to %s is listed as follows.\n", start\_date, end\_date);

homework \*p = h;

int flag = 0;

while(p)

{

if(strcmp(p->date, start\_date)>=0 && strcmp(p->deadline, end\_date)<=0) {

printHomework(p);

flag = 1;

}

p = p->next;

}

return flag;

}

有了这样的思路， 学委对作业提交情况的按时间范围查找是同一思路的， 实现方法也是大同小异。

1. 学委如何实现按升、降序对提交情况进行排序？对于不同的关键字（如：按时间升降序排序）是否需要分别写一个函数进行排序呢？既然按时间升降序进行排序，他们之间肯定有一定的规律。结合所学知识，发现选择排序使最适合这类排序问题了。其工作原理是：第一次从待排序的[数据元素](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0/715313)中选出最小（或最大）的一个元素，存放在序列的起始位置，然后再从剩余的未排序元素中寻找到最小（大）元素，然后放到已排序的序列的末尾。以此类推，直到全部待排序的数据元素的个数为零。升序（降序）不就是按最大最小值进行排序的吗？思路大致与对作业升降序查询一样。所以只需要对if语句脸面的判断条件进行修改，只需要写一个这个判断的函数，然后在这个选择排序的算法里面，把这个函数作为一个参数传进来就完成了升降排序。附部分关键代码：void selectSortSubmission(submission \*\*head, int(\*func\_p)(submission\*, submission\*)) {

submission \*p = NULL;

submission \*q = NULL;

submission \*t = NULL;

if (\*head == NULL || (\*head)->next == NULL) {

return;

}

for (p = \*head; p != NULL; p = p->next)

{

t = p;

for (q = p->next; q != NULL; q = q->next)

{

if (func\_p(q, t) > 0) {

//if (strcmp(q->course\_id, t->course\_id)>0) {

t = q;

}

}

if (t != p)

{

submission new\_t;

strcpy(new\_t.course\_id, p->course\_id);

strcpy(new\_t.course\_name, p->course\_name);

strcpy(new\_t.id, p->id);

strcpy(new\_t.submission\_time, p->submission\_time);

strcpy(new\_t.cls, p->cls);

strcpy(new\_t.remark, p->remark);

strcpy(new\_t.submission\_count, p->submission\_count);

strcpy(p->course\_id, t->course\_id);

strcpy(p->course\_name, t->course\_name);

strcpy(p->id, t->id);

strcpy(p->submission\_time, t->submission\_time);

strcpy(p->cls, t->cls);

strcpy(p->remark, t->remark);

strcpy(p->submission\_count, t->submission\_count);

strcpy(t->course\_id, new\_t.course\_id);

strcpy(t->course\_name, new\_t.course\_name);

strcpy(t->id, new\_t.id);

strcpy(t->submission\_time, new\_t.submission\_time);

strcpy(t->cls, new\_t.cls);

strcpy(t->remark, new\_t.remark);

strcpy(t->submission\_count, new\_t.submission\_count);

}

}

}

可以看出，当t=q时，则说明两条数据之间原来就是按升（降）序进行排列，的，只有当t!=q时，两条数据进行交换，使两条数据按升（降）序进行排列，从而达到想要的效果。所以只需要传入需要排序的指针，通过指针函数方式判断是升序还是降序。这是具体的调用情况。

selectSortSubmission(&submission\_list, ascendingCount);

printSubmissionInformation(submission\_list);

1. 上面两个难题是说明查找以及对数据进行排序的。而这个系统的最大特点就是

每个角色、作业、提交情况都有一样的功能，那就是增删改查这些功能，由于查找功能已经介绍，故对作业的增删改进行说明，其余功能的增删改也是一样的思想，只是传入的参数或者引用的参数不同。 附增加作业的源码：

int addHomework(homework \*\* h, homework\* t) {

homework \*p = \*h;

while(p->next) {

p = p->next;

}

homework \*new\_t;

new\_t = (homework\*)malloc(sizeof(homework));

strcpy(new\_t->course\_id, t->course\_id);

strcpy(new\_t->course\_name, t->course\_name);

strcpy(new\_t->times, t->times);

strcpy(new\_t->details, t->details);

strcpy(new\_t->date, t->date);

strcpy(new\_t->remark, t->remark);

strcpy(new\_t->deadline, t->deadline);

new\_t->next = NULL;

p->next = new\_t；

return 1;

}

由于存储用到链表，故新增作业要用到malloc动态向内存申请空间。书本也有类似代码，稍作修改即可。同理，删除和修改只需对参数修改，小修小补即可，不一一详细解释。

1. 如何对文件的读取和保存？由于大一利用C语言编写过大作业，故对此操作熟悉。只需要判断是否由该文件夹，若没有则创建，若有则直接读取或保存。附对提交信息的保存源码：

void saveSubmissionInformation(submission \*h) {

FILE \* fp;

if ((fp = fopen("submission.txt", "w")) == NULL)

{

printf("Write Error : students.txt is not existed.");

exit(-1);

}

submission \*p = h;

while (p) {

fprintf(fp, "%s\t%s\t%s\t%s\t%s\t%s\t%s\n", p->id, p->cls, p->course\_name,p->course\_id, p->submission\_time, p->submission\_count, p->remark);

p = p->next;

}

fclose(fp);

}

这是对文件的读写源码：

void readSubmission(submission \*\*head) {

FILE \* fp;

if((fp = fopen("submission.txt","r")) == NULL)

{

printf("Read Error : submission.txt is not existed.");

exit(-1);

}

\*head = (submission\*)malloc(sizeof(submission));

submission \*p = \*head, \*tail;

char str[MAX\_SUBMISSION\_STRING\_LEN];

while(fscanf(fp,"%s",str)!=EOF) {

tail = (submission\*)malloc(sizeof(submission));

strcpy(tail->id, str);

fscanf(fp, "%s", tail->cls);

fscanf(fp, "%s", tail->course\_name);

fscanf(fp, "%s", tail->course\_id);

fscanf(fp, "%s", tail->submission\_time);

fscanf(fp, "%s", tail->submission\_count);

fscanf(fp, "%s", tail->remark);

tail->next = NULL;

p->next = tail;

p = p->next;

}

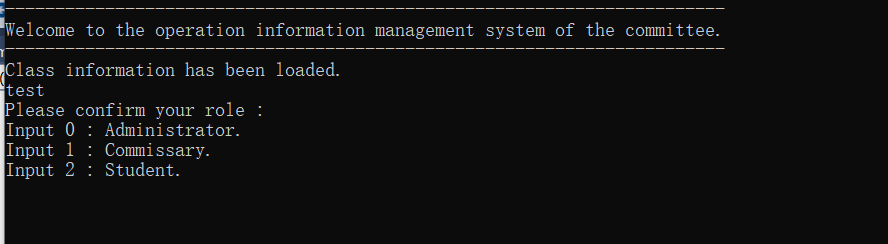
\*head = (\*head)->next;

}

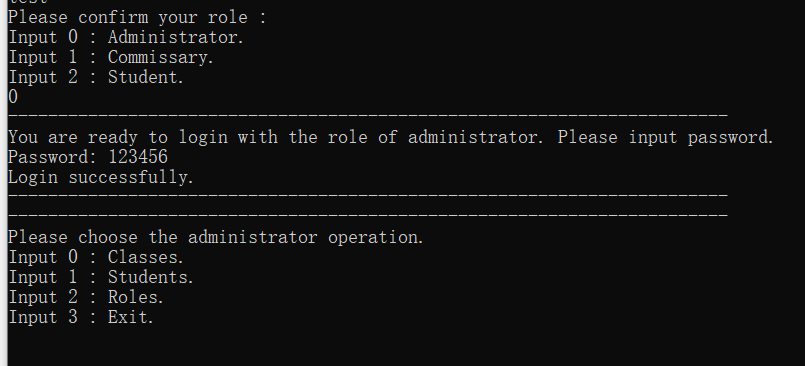
## 4.程序测试说明

从上面的关键难点逐一进行突破，终于实现了学委作业管理系统。下面附上完成后的系统效果图：

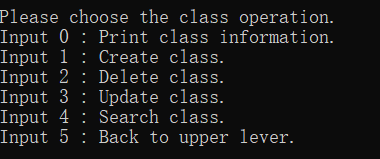
系统登陆界面：



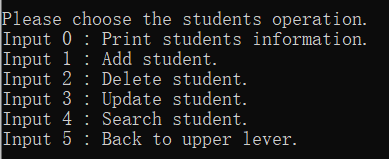
管理员登陆界面：



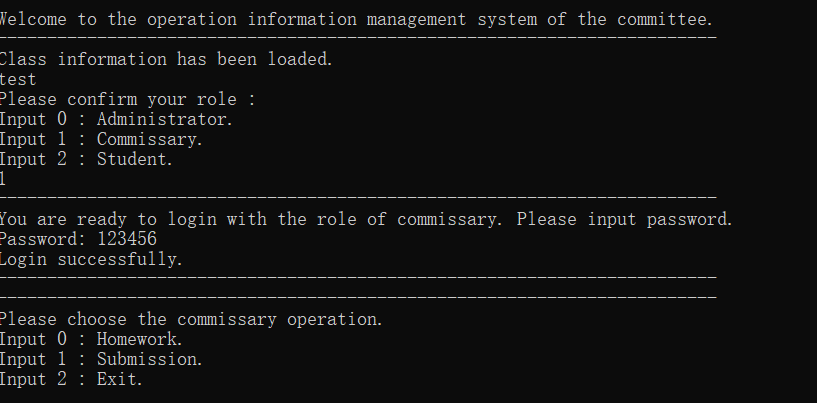
管理员对班级操作界面：



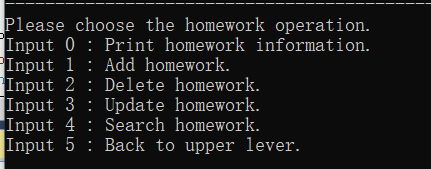
管理员对学生操作界面：



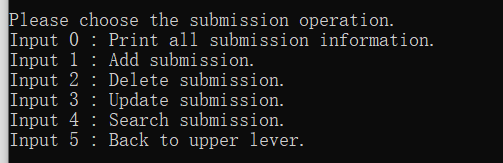
学委登陆界面：



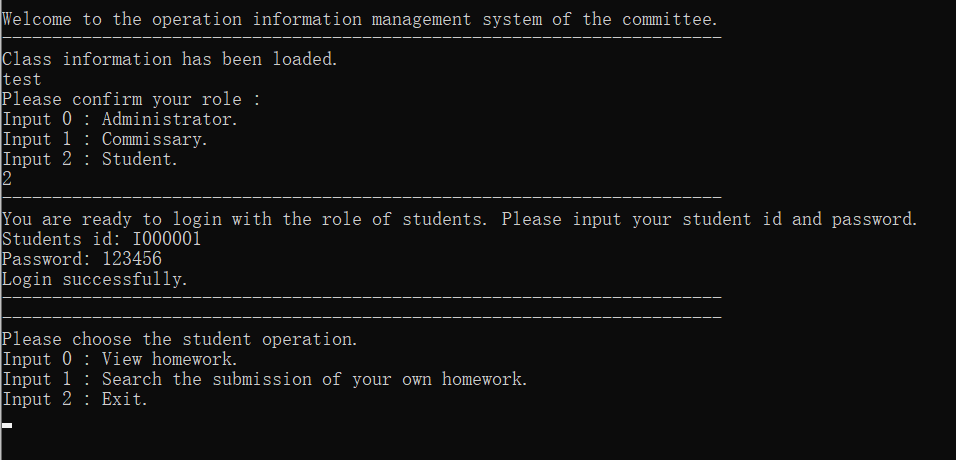
学委对作业操作界面：



学委对提交情况操作界面：



学生登录及操作界面：



## 三、总结

结合本次开发历程，收获颇多。一是自己遇到难题的时候不再退缩寻找解决方法，这是一个较大的进步。第二是先懂得题目叫做什么，一一列举出来，按照这个结果进行开发，过程变得游刃有余。第三是开发思维不再单一变得多样，一个问题不是只有一个解决方法，把自己的思维局限与课本内是不会进步的，正所谓方法总比困难多也不无道理的。同时，由于对程序源码编写更美观，通过逐级调用思想令代码更简洁，不过要查看源码则需要查找。另外，这是我第一次全英编写程序，对以后的编程也有模范作用。

# 四、参考文献

[1] 《数据结构》 严慧敏 吴伟民 主 编 清华大学出版社