



# 信息与软件工程学院

## 项目报告

课程名称：\_\_\_\_程序设计项目实践（PBLF）\_\_\_\_

学 期：\_\_\_\_2023-2024 第 1 学期\_\_\_\_

项目名称：\_\_\_\_排班系统\_\_\_\_

授课教师：\_\_\_\_吴劲\_\_\_\_

序号	学号	姓名
1（组长）	2023090902018	石旭民
2	2023090901017	杨智钧
3	2023090902014	何浩民
4	2023090901021	梁原硕
5		

# 目录

- 1 项目简介 ..... 3
  - 1.1 考核方式 ..... 3
  - 1.2 项目题目及内容简介 ..... 3
  - 1.3 项目组成员与分工 ..... 4
- 2 需求分析 ..... 4
  - 2.1 选题的依据 ..... 4
  - 2.2 功能需求 ..... 5
- 3 系统设计 ..... 6
  - 3.1 总体设计 ..... 6
  - 3.2 模块设计 ..... 6
- 4 系统实现 ..... 8
  - 4.1 主函数 ..... 8
  - 4.2 其他函数 ..... 9
- 5 测试 ..... 13
- 6 总结 ..... 14
  - 姓名-学号 ..... 18

# 1 项目简介

## 1.1 考核方式

总成绩 = 项目和项目文档成绩(40%) + 汇报幻灯片成绩(20%)  
+ 表达能力(20%) + 团队合作(20%)

## 1.2 项目题目及内容简介

### 1 问题描述

某单位有 7 名保安人员，要求每个人在一星期中可以休息一天。每名保安可以自行选择自己想要的休息日，而对于休息日的选择既可以是某一天，也可以是某几天中的其中一天。当 7 名保安依次输入自己想要的休息日后，系统将会给出可能的值班安排。若系统给出的值班安排是 0 种，证明系统对 7 名保安的选择无法作出排班，需要重新输入。

要求打印轮休的所有可能方案。当然使每个人都满意，例如每人可以选择的休息日如下：

钱：星期一、星期六

孙：星期三、星期日

李：星期五

周：星期一、星期四、星期六

吴：星期二、星期五

陈：星期三、星期六、星期日

运行结果：

排班表： 1

赵 钱 孙 李 周 吴 陈

星期四 星期一 星期五 星期六 星期三 星期二 星期日

排班表： 2

赵 钱 孙 李 周 吴 陈

星期四 星期一 星期日 星期五 星期六 星期二 星期三

排班表： 3

赵 钱 孙 李 周 吴 陈

星期四 星期六 星期三 星期一 星期五 星期二 星期日

排班表： 4

赵 钱 孙 李 周 吴 陈

星期四 星期六 星期日 星期五 星期一 星期二 星期三

## 2 功能要求

代码要能提供以下几个基本功能。

(1) 用户登录：

- 新用户可以注册，旧用户直接登录。

(2) 排班表：

- 实现选择并存储排班表到排班表文件中。
- 维护排班表文件基本信息，实现增加、修改、查询、删除排班表记录的功能。
- 打卡功能。

## 1.3 项目组成员与分工

石旭民：整合代码，编写文档，做 PPT。

杨智钧：核心代码编写，实现排班。

梁原硕：用户信息管理。

何浩民：排班表信息管理。

# 2 需求分析

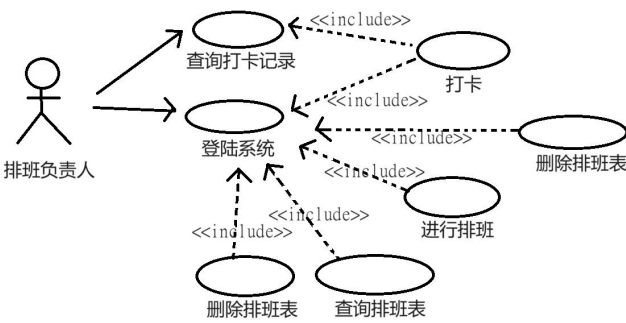
## 2.1 选题的依据

在生活中，常常会遇到轮班冲突的情况，大家商量不好的话就会大打出手，后果非常的严重。如果是依靠电脑来完成排班，就可以保证公平，而且非常的方便。于是我们就决定做一个排班系统来实现这一功能。由于我们小组是四个臭皮匠组成的，所以在商量选题的时候都没有什么想法。一来是没有经验不知道有什么选题，而来也不知到要实现我们的选题到底有多难，所以我们就在网络上查找了一下有什么适合我们做的项目，后来就发现这个排班系统好像挺合适，实用而

且应该不会很难，除了文件操作比较生疏外其他的知识都还算掌握的比较好。于是就选择了这个选题。

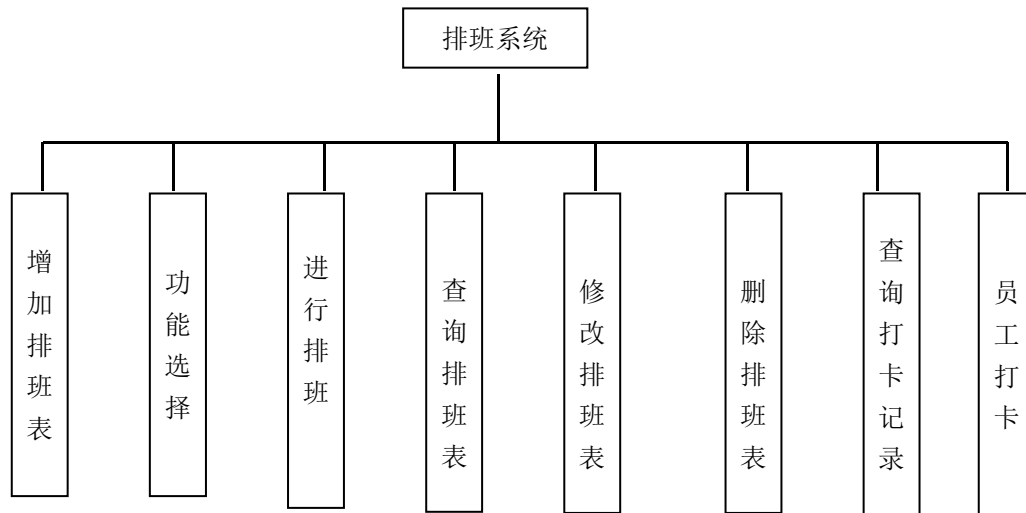
## 2.2 功能需求

使用 UML 用例图说明系统功能



## 3 系统设计

### 3.1 总体设计



### 3.2 模块设计

分为四个模块来完成项目，包括排班表信息管理（`information_control.c`），用户信息管理(`user_control.c`)，实现排班(`mainstream.c`)和主菜单(`menu_choice.c`)

各种库函数的头文件，宏定义，结构体

all.h

```
void printSchedule(int* n);
void schedule_control(char* username, int* n, char name[NUM_GUARDS][5]);
void saveScheduleToFile(int n, char name[7][5], int desire[7][7], int tmp[7], int day[7]);
void loadSchedule(int(*guards)[NUM_DAYS], char** guard_names);
void changeSchedule(char name[NUM_GUARDS][5], int* n);
void push();
```

information\_control.h

```
void arranging(char* username, char* use);
void ask_desire(char name[7][5], int desire[7][7]);
void sort(int day[7], int tmp[7], int* n, int desire[7][7], int order);
void fail(int* n, char name[7][5], int desire[7][7], int tmp[7], int day[7], char* use);
void add_arrage(int* n, char name[7][5], int desire[7][7], int tmp[7], int day[7], char* use);
void user_menu(int* n, char name[7][5], int desire[7][7], int tmp[7], int day[7], char* use);
void esc(int* n, char name[7][5], int desire[7][7], int tmp[7], int day[7], char* use);
```

mainstream.h

```
void loginUser(User users[MAX_USERS]);
void registerUser(User users[MAX_USERS]);
```

user\_control.h

```
#include "all.h"
#include "information_control.h"
#include "mainstream.h"
#include "user_control.h"
void menu();
int main();
```

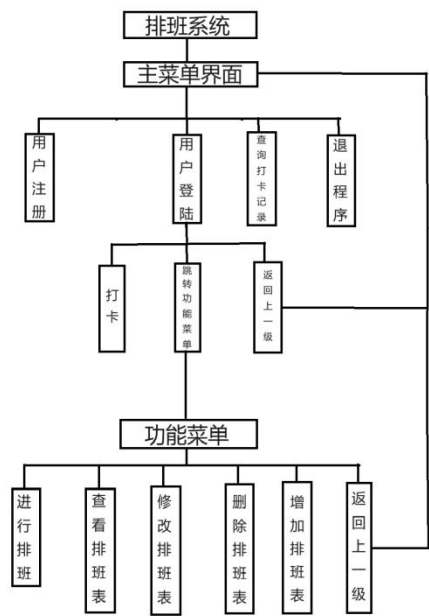
menu\_choice.c

Menu\_choice.c 显示主菜单并调用 user\_control.c 中的 loginUser 和 registerUser 函数进行用户登录和注册操作。登陆成功之后调用 arranging 函数将操作排班表要用到的变量进行初始化，并将变量传给 information\_control.c 中的 schedule\_control 函数进行选择。选择查看排班表则调用 print Schedule 函数进行打印；选择修改排班表则调用 changeSchedule 函数在备份排班表文件中生成新的排班表，再用 push 函数把备份排班表的内容复制到原文件中；选择删除排班表则删除排班表文件；选择进行排班则返回 arranging 函数进行下一步操作。返回之后，arranging 函数首先调用 ask\_desire 函数记录员工意愿，再调用 sort 函数进行排班，最后调用 fail 函数检查排班是否失败。

# 4 系统实现

## 4.1 主函数

用程序流程图说明实现思路



关键代码分析



```

void ask_desire(char name[7][5], int desire[7][7])
{
    int i, j, k;
    char week[7][10] = { "一", "二", "三", "四", "五", "六", "日" };
    char ch[30];
    printf("请输入下列人员的休息日: \n");
    for (i = 0; i < 7; i++)
    {
        k = 0;
        printf("%s:", name[i]);
        fflush(stdin);
        scanf("%s", ch);
        for (j = 0; j < 7; j++)
        {
            if (strstr(ch, week[j]) != NULL)
            {
                desire[i][k] = j + 1; //存储意愿 人物/日期
                k++;
            }
        }
    }
}

```

首先是询问每个保安想要在哪一天休息，这里就用了这样一段代码。其主要用于获取员工的休息日和排班偏好。它接收两个参数：**name** 是一个包含 7 个员工姓名的二维字符数组，**desire** 是一个用于存储排班偏好的二维整数数组。在函数内部，首先定义一个表示星期的字符串数组 **week**，包含了每个星期的名称。然后，通过循环遍历每个员工，依次询问他们的休息日。接下来，通过嵌套循环遍历每个星期，使用 **strstr** 函数判断用户输入的字符串中是否包含当前星期的名称。如果包含，则将该日期的索引加 1 存入 **desire** 数组中，并将用于记录排班偏好的索引值 **k** 增加 1。最后，完成对所有员工的询问后，**desire** 数组中存储的就是每个员工对应的排班偏好。

```

void sort(int day[7], int tmp[7], int* n, int desire[7][7], int order)
{
    int i, j;
    if (order == 7)
    {
        (*n)++;
        FILE* fp = fopen(TEMP_FILE, "a+");
        printf("\n排班表%d\n", *n);
        fprintf(fp, "排班表%d\n", *n);
        printf("赵\t钱\t孙\t李\t周\t吴\t陈\t\n");

        for (i = 0; i < 7; i++)
        {
            switch (tmp[i]) {
                case 1:printf("星期一\t"); fputs("星期一\t", fp); break;
                case 2:printf("星期二\t"); fputs("星期二\t", fp); break;
                case 3:printf("星期三\t"); fputs("星期三\t", fp); break;
                case 4:printf("星期四\t"); fputs("星期四\t", fp); break;
                case 5:printf("星期五\t"); fputs("星期五\t", fp); break;
                case 6:printf("星期六\t"); fputs("星期六\t", fp); break;
                case 7:printf("星期日\t"); fputs("星期日\t", fp); break;
                default:break;
            }
        }
        printf("\n");
        fputs("\n", fp);
        fclose(fp);
        return;
    }
}

```

```

    for (j = 0; j < 7 && desire[order][j]; j++)
    {
        if (day[desire[order][j] - 1] != 0) //该时间被占有
        {
            continue;
        }

        day[desire[order][j] - 1] = 1;
        tmp[order] = desire[order][j];

        sort(day, tmp, n, desire, order + 1); //递归

        day[desire[order][j] - 1] = 0;
        //递归后将这一天置0去遍历这一天不选的情况
    }
}

```

然后是排班的实现。这两段代码是 `sort` 函数的实现，用于生成排班表。它接收五个参数：`day` 是一个表示每天是否被占用的数组，`tmp` 是一个临时数组用于保存当前排班表的状态，`n` 是一个指向整数的指针，用于记录已生成的排班表数量，`desire` 是一个表示员工对每天的排班偏好的二维数组，`order` 表示当前正在进行排班的员工序号。在函数内部，首先判断如果已经排完了所有员工（即 `order == 7`），则代表当前的 `tmp` 数组中存储了完整的一份排班表，此时将排班表数量增加 1，并打印出当前排班表。通过 `fopen` 函数打开一个名为 `TEMP_FILE` 的文件，并将排班表的信息写入文件。在每次循环中，根据 `tmp` 数组中存储的员工排班情况，使用 `switch` 语句将日期转换成星期，并打印出每个员工的排班情况。最后，关闭文件并返回函数。该函数使用递归实现全排列，并在每次递归到最后一层时，即完成一份排班表时，进行输出和保存操作。

这段 for 循环代码是在递归函数中的一个关键部分，它用于遍历员工 order 的排班偏好，并尝试安排员工在他们偏好的时间休息。for (j = 0; j < 7 && desire[order][j]; j++)：这是一个循环语句，用于遍历员工 order 的排班偏好。循环的条件是 j < 7，即要求 j 小于 7；以及 desire[order][j] - 1 != 0：这行代码用于检查当前正在考虑的日期是否已经被占用。desire[order][j] 表示员工 order 在第 j 天的排班偏好，day[desire[order][j] - 1] 则表示这一天的占用情况（0 表示未被占用，1 表示已被占用）。er[j] 不为 0，表示员工 order 在第 j 天有排班偏好。这个循环会依次检查员工 order 在每一天的排班偏好。if (day[desire[order][j] - 1] != 0)：如果这一天已经被占用，则执行 continue 跳过当前循环，继续考虑下一天的排班偏好。如果这一天没有被占用，那么接下来的操作将安排员工在这一天工作：day[desire[order][j] - 1] = 1；：将这一天标记为占用。tmp[order] = desire[order][j]；：将员工 order 安排在这一天工作，将排班情况写入 tmp 数组。sort(day, tmp, n, desire, order + 1)；：进行递归调用，处理下一个员工的排班安排。

day[desire[order][j] - 1] = 0；：在递归返回后，将占用的这一天重新标记为未占用，以便尝试其他排班方式。这样可以确保在尝试不同排班方案时，不会受之前排班的影响。总之，这段代码通过循环遍历员工 order 的排班偏好，尝试安排员工在他们偏好的时间工作，并对每一种可能的排班情况进行递归处理。

```
void fail(int* n, char name[7][5], int desire[7][7], int tmp[7], int day[7], char* use)
{
    int i, j;
    if (*n == 0)
    {
        int xq[7] = { 0, 0, 0, 0, 0, 0, 0 };
        for (i = 0; i < 7; i++)
        {
            for (j = 0; j < 7; j++)
            {
                if (desire[i][j] == 0) continue;
                if (desire[i][j] == 1) xq[1] = 1;
                if (desire[i][j] == 2) xq[2] = 1;
                if (desire[i][j] == 3) xq[3] = 1;
                if (desire[i][j] == 4) xq[4] = 1;
                if (desire[i][j] == 5) xq[5] = 1;
                if (desire[i][j] == 6) xq[6] = 1;
                if (desire[i][j] == 7) xq[0] = 1;
            }
        }

        if (xq[0] == 0) printf("星期日");
        if (xq[1] == 0) printf("星期一");
        if (xq[2] == 0) printf("星期二");
        if (xq[3] == 0) printf("星期三");
        if (xq[4] == 0) printf("星期四");
        if (xq[5] == 0) printf("星期五");
        if (xq[6] == 0) printf("星期六");
        printf("无法排班\n");
        esc(n, name, desire, tmp, day, use);
    }
}
```

这段代码的主要功能是检测排班是否失败。首先，判断是否所有员工已经被

安排休息，即 $*n == 0$ 。如果是，则进入下一步检测排班是否合法。在接下来的循环中，遍历所有员工的排班偏好（即 `desire` 数组），并将每个员工可以工作的星期标记在一个长度为 7 的数组 `xq` 中。其中，`xq[0]` 表示星期日，`xq[1]~xq[6]` 分别表示星期一到星期六。如果员工在某一天有排班偏好，则对应的数组元素置为 1。最后，如果 `xq` 数组中存在某一天所有员工都无法工作（即某一天所有元素都为 0），则认为排班失败，输出错误信息并结束程序。

## 4.2 其他函数

```
void clock_in(User* use); //打卡

void clock_record(); //查询打卡记录

void loginUser(User users[MAX_USERS]); //登录

void registerUser(User users[MAX_USERS]); //注册

void printSchedule(int* n); //打印排班表

void schedule_control(char* username, int* n, char name[NUM_GUARDS][5]);
//排班表信息管理主函数

void saveScheduleToFile(int n, char name[7][5], int desire[7][7], int tmp[7], int
day[7]); //增加排班表

void loadSchedule(int(*guards)[NUM_DAYS], char** guard_names); //加载排
班表

void changeSchedule(char name[NUM_GUARDS][5], int* n); //修改排班表

void push(); //修改好的排班表拷贝回原文件

void arranging(char* username, char* use);

Void      ask_desire(char      name[7][5],      int      desire[7][7]);
//询问意愿

void sort(int day[7], int tmp[7], int* n, int desire[7][7], int order);
//实现排班

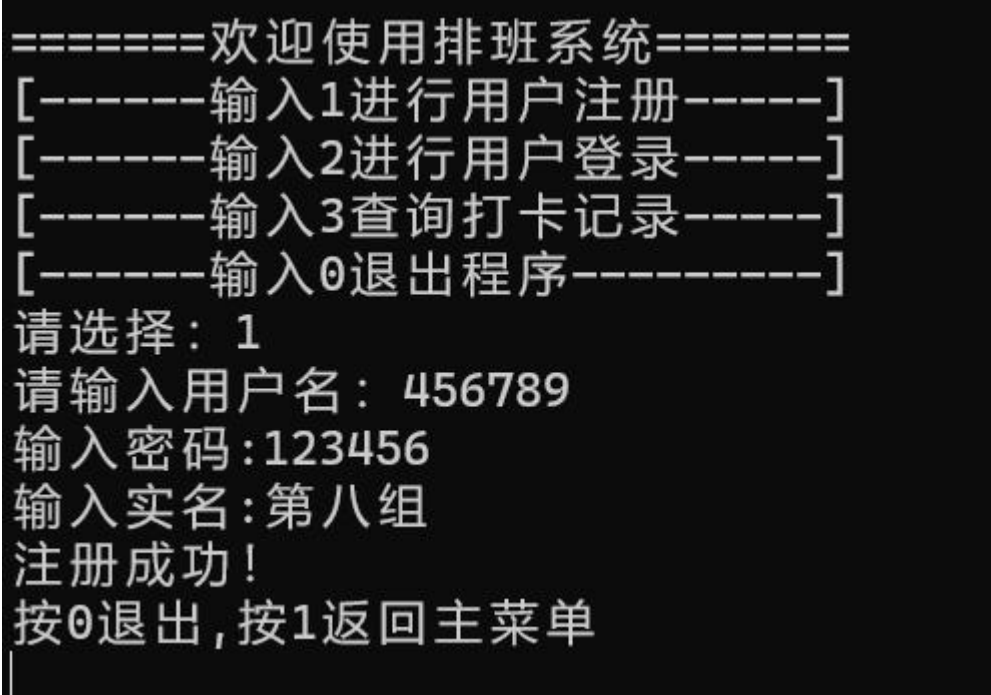
void fail(int* n, char name[7][5], int desire[7][7], int tmp[7], int day[7], char*
use); //检测排班是否失败

void add_arrage(int* n, char name[7][5], int desire[7][7], int tmp[7], int day[7],
char* use); //增加排班
```

```
void user_menu(int* n, char name[7][5], int desire[7][7], int tmp[7], int day[7],  
char* use);          //用户功能菜单  
  
void esc(int* n, char name[7][5], int desire[7][7], int tmp[7], int day[7], char*  
use);
```

## 5 测试

### 1. 注册



```
=====欢迎使用排班系统=====  
[-----输入1进行用户注册-----]  
[-----输入2进行用户登录-----]  
[-----输入3查询打卡记录-----]  
[-----输入0退出程序-----]  
请选择：1  
请输入用户名：456789  
输入密码：123456  
输入实名：第八组  
注册成功！  
按0退出，按1返回主菜单  
|
```

### 2. 登录

```
=====欢迎使用排班系统=====
[-----输入1进行用户注册-----]
[-----输入2进行用户登录-----]
[-----输入3查询打卡记录-----]
[-----输入0退出程序-----]
请选择：2
请输入用户名：456789
请输入密码：123456
登录成功，按1进行打卡，按2跳转到用户菜单，按0返回上一级
请选择：|
```

### 3. 打卡

```
=====欢迎使用排班系统=====
[-----输入1进行用户注册-----]
[-----输入2进行用户登录-----]
[-----输入3查询打卡记录-----]
[-----输入0退出程序-----]
请选择：2
请输入用户名：456789
请输入密码：123456
登录成功，按1进行打卡，按2跳转到用户菜单，按0返回上一级
请选择：1

第八组于2023-12-13 18:3:26进行打卡考勤

请按任意键继续...|
```

### 4. 查询打卡记录

```
=====欢迎使用排班系统=====
[-----输入1进行用户注册-----]
[-----输入2进行用户登录-----]
[-----输入3查询打卡记录-----]
[-----输入0退出程序-----]
请选择：3
石于2023-12-10 13:7:57进行打卡考勤

石于2023-12-12 15:54:52进行打卡考勤

石于2023-12-13 11:35:37进行打卡考勤

第八组于2023-12-13 18:3:26进行打卡考勤

按0退出，按1返回上一级，按2清空记录
|
```

## 5.用户菜单

```
|| ==      用户菜单      == ||
|| == 输入1创建排班表 == ||
|| == 输入2查询排班表 == ||
|| == 输入3修改排班表 == ||
|| == 输入4删除排班表 == ||
|| == 输入5增加排班表 == ||
|| == 输入6退回上一级 == ||
请选择：|
```

## 6. 进行排班

```

=====欢迎来到排班模块=====
[-----Press 1 : 进行排班-----]
[-----Press 2 : 主菜单-----]
[-----Press 0 : 退出系统-----]
请输入您的选择: 1
请输入下列人员的休息日:
赵:星期二星期四
钱:星期一星期六
孙:星期三星期日
李:星期五
周:星期一星期四星期六
吴:星期二星期五
陈:星期三星期六星期日

排班表1
赵    钱    孙    李    周    吴    陈
星期四 星期一 星期三 星期五 星期六 星期二 星期日

排班表2
赵    钱    孙    李    周    吴    陈
星期四 星期一 星期日 星期五 星期六 星期二 星期三

排班表3
赵    钱    孙    李    周    吴    陈
星期四 星期六 星期三 星期五 星期一 星期二 星期日

排班表4
赵    钱    孙    李    周    吴    陈
星期四 星期六 星期日 星期五 星期一 星期二 星期三
请按任意键继续. . . |

```

## 7. 增加排班表



请输入下列人员的休息日：

赵：星期一  
钱：星期二  
孙：星期三  
李：星期四  
周：星期五  
吴：星期六  
陈：星期日

排班表格已保存到文件 temp.txt 中。

```
|| ==      用户菜单      == ||
|| == 输入1创建排班表 == ||
|| == 输入2查询排班表 == ||
|| == 输入3修改排班表 == ||
|| == 输入4删除排班表 == ||
|| == 输入5增加排班表 == ||
|| == 输入6退回上一级 == ||
请选择：|
```

排班表(休息日):

	赵	钱	孙	李	周	吴	陈
排班表1	星期四	星期一	星期三	星期五	星期六	星期二	星期日
排班表2	星期四	星期一	星期日	星期五	星期六	星期二	星期三
排班表3	星期四	星期六	星期三	星期五	星期一	星期二	星期日
排班表4	星期四	星期六	星期日	星期五	星期一	星期二	星期三
排班表5	星期一	星期二	星期三	星期四	星期五	星期六	星期日

```
|| ==      用户菜单      == ||
|| == 输入1创建排班表 == ||
|| == 输入2查询排班表 == ||
|| == 输入3修改排班表 == ||
|| == 输入4删除排班表 == ||
|| == 输入5增加排班表 == ||
|| == 输入6退回上一级 == ||
请选择：|
```

## 6 总结

### 姓名-学号

你的总结

学到了什么？

痛点和难点

自己的贡献

如何与他人合作？

石旭民-2023090902018

我的总结：在这次之前我们从来没有做过这样一个需要合作分工的，较大的项目。虽然成品十分的简单，代码也比较挫，可移植性不高，但在这个过程中我们第一次正真意义上认识到了团队合作的概念，并收获了满满的成就感，而且为之后合作写项目提供了宝贵的经验。

学到了什么：通过这次经历，我对一个项目中各个模块之间的联系的理解变得更加深入了，并在帮助组员修改代码以及最后整合代码的过程中领略到了别人写代码的风格以及一些实用的写法，比如说利用 `fflush (stdin)` 清空标准输入流的缓存，避免输入粘连；利用命令行指令 `cls` 清空屏幕上的内容，达到界面切换的效果。还学到了用 C 语言修改文件中指定部分的方法。同时也积累了团队合作的经验。

痛点和难点：首先是代码关联性的问题。这次项目中我负责的是代码的整合，本以为会是很轻松的工作，但是，由于我没有经验，我没有叫我们小组的成员在开写之前先坐下来商量一下，比如说先把各种宏定义，结构体什么的规定好，什么数据存到哪个文件中，自己要用到什么函数，函数是什么功能，需要什么参数，以及整个函数的流程，用户的需求怎么实现。结果就导致之后组员们写出来的代码有点牛头不对马嘴，参数多而繁杂；结构体，全局变量重定义，函数功能重复情况严重，排班表管理模块管理的排班表的格式和实现排班模块的不一样。还有很多功能重复但名称不一样的文件，使得我们在最后花了很多时间再来统一。其次就是编程能力的问题，这个选题是在网上发现的，本来很多的额外功能我们都打算实现一下，可惜最后发现超出我们的能力范围了，因为时间精力等原因都没搞成。

我的贡献：就是把组员们的代码整合起来，顺便加深对程序的理解，以便编写文档。由于我们事先没有商量好，流程图都是把代码跌跌撞撞地整合好之后才对着程序的运行结果画出来的，非常的失败，不过以后就有这方面的经验了。

如何与他人合作：在开始动手做之前，一定要坐下来好好商量，安排好各项事宜，在动手做的过程中也要及时与他人沟通，有不清楚的地方及时询问负责相关模块的人。团队合作要想达到超过个人的效果，那他首先要像一个体，不能像脑瘫一

样，走在路上四肢各走各的。

## 梁原硕-2023090901021

总结：经过第一次的项目小组合作后，收获颇多，也见到了很多不足之处，我负责用户登录以及打卡部分的代码，相较而言这部分难度并不是很高，对于我来说却是个很好的机会，能更好的体会指针、数组等结构的运用特点，而且因为我自身并没有很好的基础，这个难度的代码也正好能够作为我的考验，首先在编写用户注册和登录部分的代码时，我先决定编写主要代码，我在其中使用了很多数组在完成储存数据，同时尽量避免了指针的使用，因为在编写过程中我曾使用过指针，结果出现了报错并且经过查询后发现我并不能够很好的理解其中的错误，这使我意识到我对于指针还是不够熟练，但这并不是我遇到的最大的难题。在完成基本代码后我意识到要进行文件储存数据的代码编写，这也是我的一个痛点之一，于是本着工欲善其事必先利其器的原则，我在 B 站观看了文件储存的教学，结合雨课堂的教学，最终我还是攻克了这一难关，但是做项目从来不是一路顺风的，在我编写打卡代码时，我本以为这部分会比用户部分简单，但我没想到在编写是一个#发生了错误，更棘手的是 `vscode` 并没有因此报错，可以正常运行但功能失效，我并没有处理这种情况的经验，于是我再次上网寻找可能的原因，可惜的是，我并没有成功，同时由于时间有限，我只能放弃这一部分的代码，重新开始编写新的代码，而这一次我决定利用上指针以简化结构，上一次的错误也与代码冗长有一定关联，这使得我无法定位报错原因，于是我用上了指针，十分小心的完成了这一部分的代码编写，但是到了最后项目各部分合成的时候还是出现了问题，部分代码并不能很好的衔接，不过这些问题不再只有我一个人面对，而是我们一起在组长的带领下进行修改，我们将头文件提取，将变量统一，经逻辑捋顺，最终完成了项目，尽管并没有达到完美的地步，但我还是很满意也很欣喜，毕竟对于我而言，这是我第一次参与项目的编写，也是第一次进行 C 语言的实践，小组的大家都很好，十分和气，也乐意提供帮助，组长更是包揽了很多关键的工作，所以这次合作可谓十分舒畅，至少在我过去的合作中算得上体验良好的一次，以上便是我的总结

## 何浩民-2023090902014

通过参与这个项目，作为一个 C 语言初学者，我学到了编写高效且易于维护的代码的重要性。

与团队成员密切合作过程中，我负责用户信息管理模块。我深刻体会到了团队协作的重要性，以及倾听和妥善处理分歧的重要性，比如给参数的命名，以前我可能只会命名为 a, b 之类，代码也不会加上注释，导致在团队整合的时候队友看不明白。

在设计这个项目过程中，作为一个 C 语言初学者，感觉到程序设计的不易，经验不足，很多时候都不知道该怎么做，需要寻求队友的帮助。并且自己写的程序又总是有莫名其妙的报错，又或者是忘记打分号这种细节错误。经历很多挫折，最终成功实现了这个用户信息管理模块。这个项目带给我宝贵的经验，对我以后的

学习和工作有着积极的影响。

## 杨智钧-2023090901017

对我来说，在这次程序项目设计中，我主要是学到了团队协作的能力，虽然我写的代码比较简单，但过程中还是出现了许多杂七杂八的问题，其中有很多都是我通过询问组长解决的，可见，团队的力量是无穷的。关于过程中的痛点与难点，主要有两点，其实这两点也可以归为一点，由于组员之间的交流不及时，不清晰明确，导致我们各自编写的代码不匹配，给项目的推进工作带来了许多麻烦。然后是我的贡献，前面讲了，我主要是负责编写实现排版系统的递归算法。最后是如何与他人合作，在我看来，建立有效的团队分工和角色职责以及成员之间明确的沟通是小组进行有效合作的前提。