云南大学软件学院 实验报告

课程： 软件工程概论 学期： 2024 - 2025学年 第一学期 任课教师： 莫启

专业： 软件工程 学号： 20221120044 姓名： 唐嘉骏 成绩：

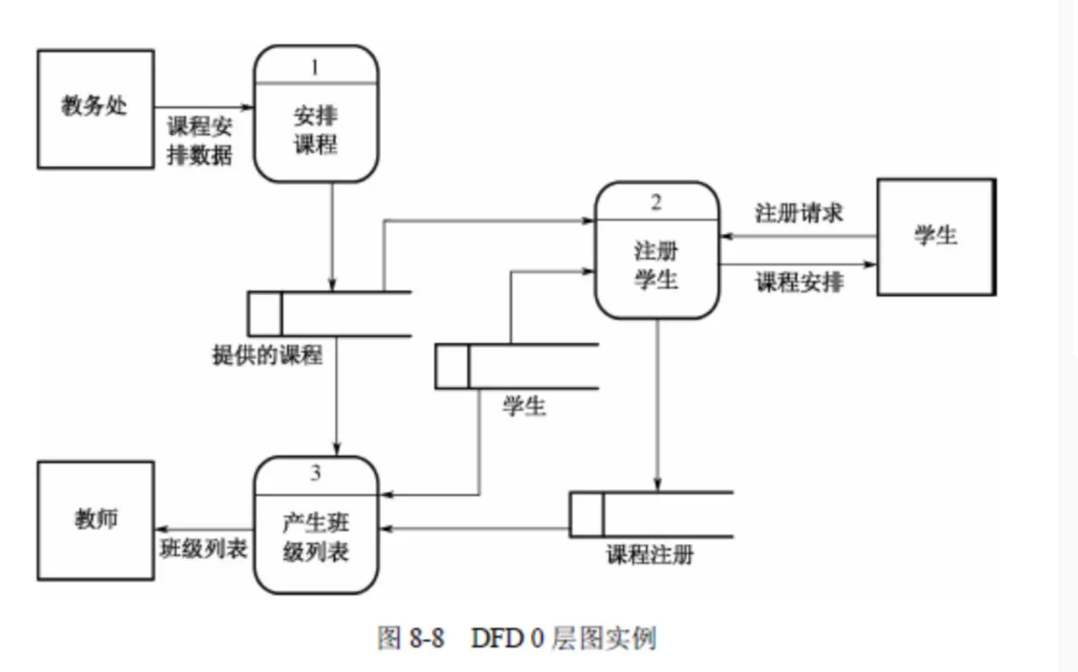
**实验1**

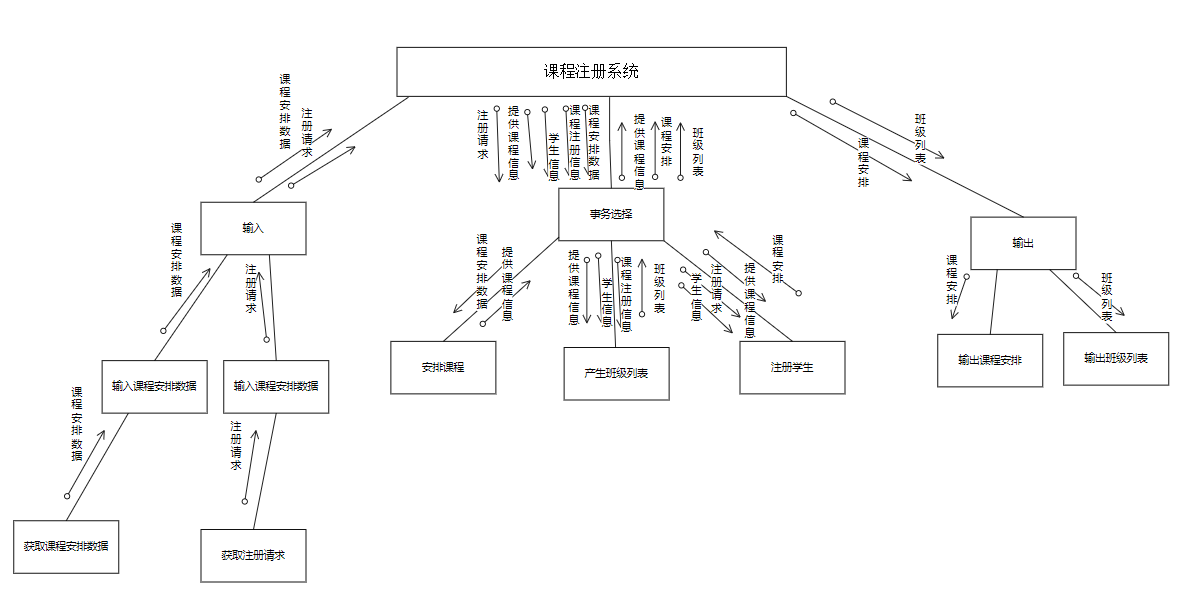
请根据如下课程注册系统的数据流图：

（1）设计其对应体系结构，以结构图绘制（用Visio或draw.io绘制）；

（2）基于生成的结构图，对其进行结构化程序设计，即：1）为结构图中每个模块确定采用的算法，并采用某种适当工具（如程序流程图）给出清晰的描述；2）确定每一模块使用的数据结构；3）确定模块接口的细节（外部接口、内部接口、用户界面等）；4）写出详细设计规格说明书。

（3）对其进行简单开发，包括代码实现及测试（单元测试课集成测试）等部分。



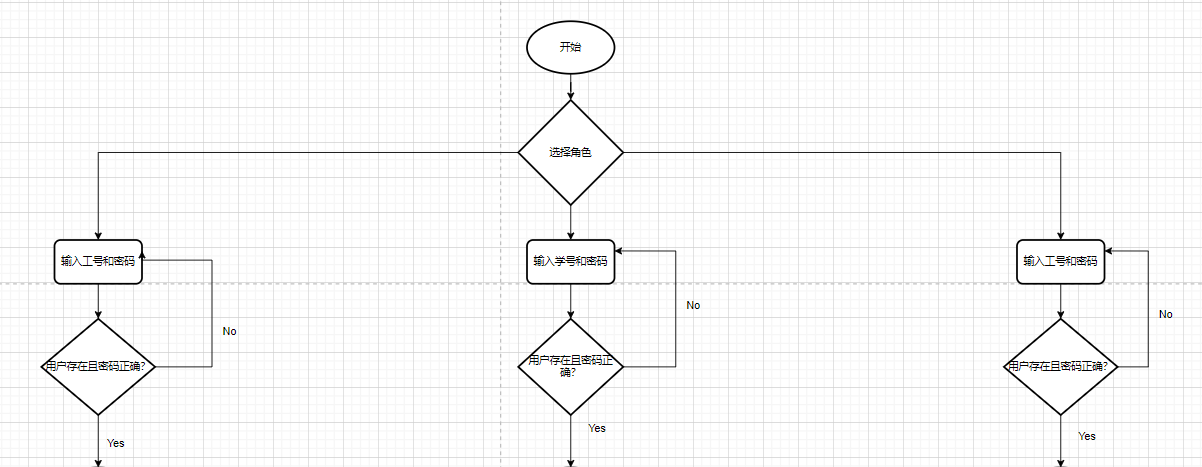
1. 对应结构图设计如下：

（2）

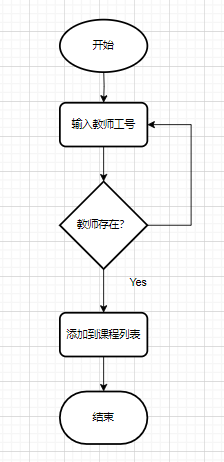
* 1. 每个模块确定算法如下：

1. 登录模块

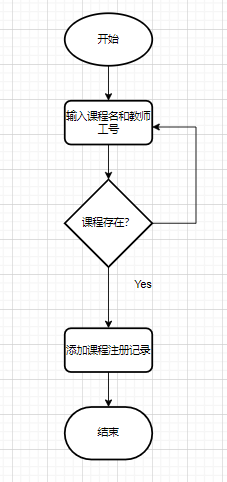
算法：通过循环遍历对应的角色数组，如果存在某一个记录与输入对应，则成功登录，返回该记录，否则返回学号/工号/密码错误。



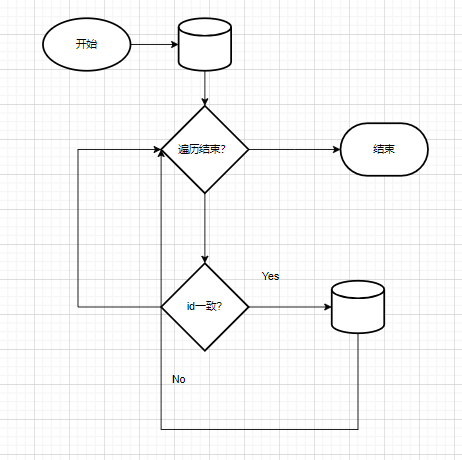
1. 新增课程模块：通过循环遍历教师列表，如果存在某一个记录的工号与输入对应，则将新的课程添加到课程列表中，否则提示错误。



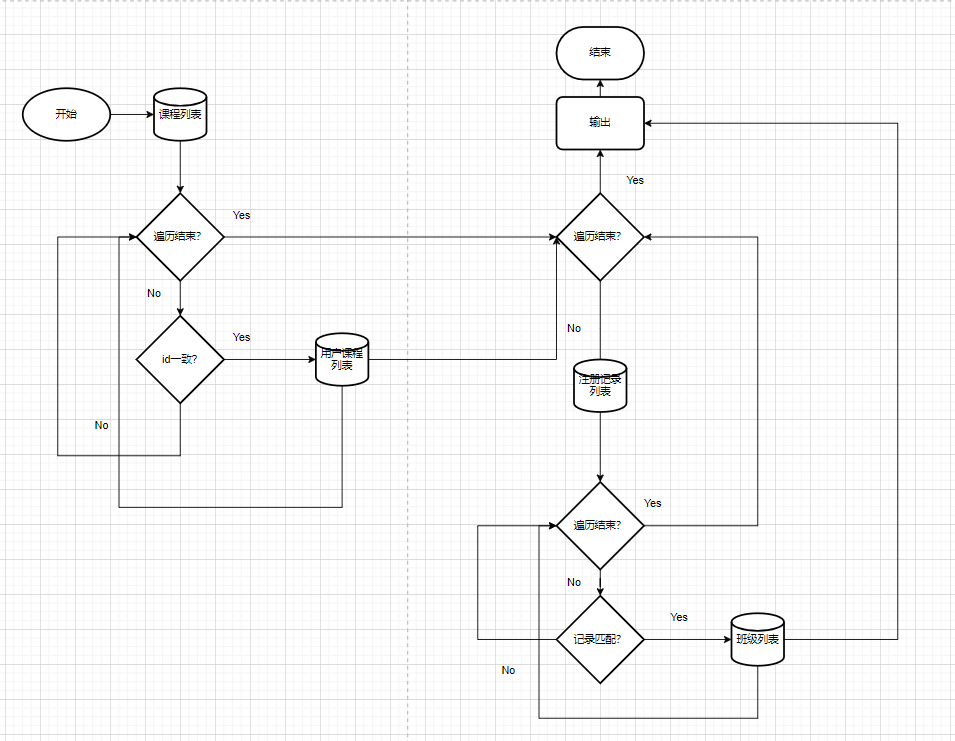
1. 注册课程模块:通过循环遍历课程列表，如果存在某一个记录的课程名与教师工号与输入对应，则将用户与该课程合成一条记录追加到记录列表，否则提示错误。



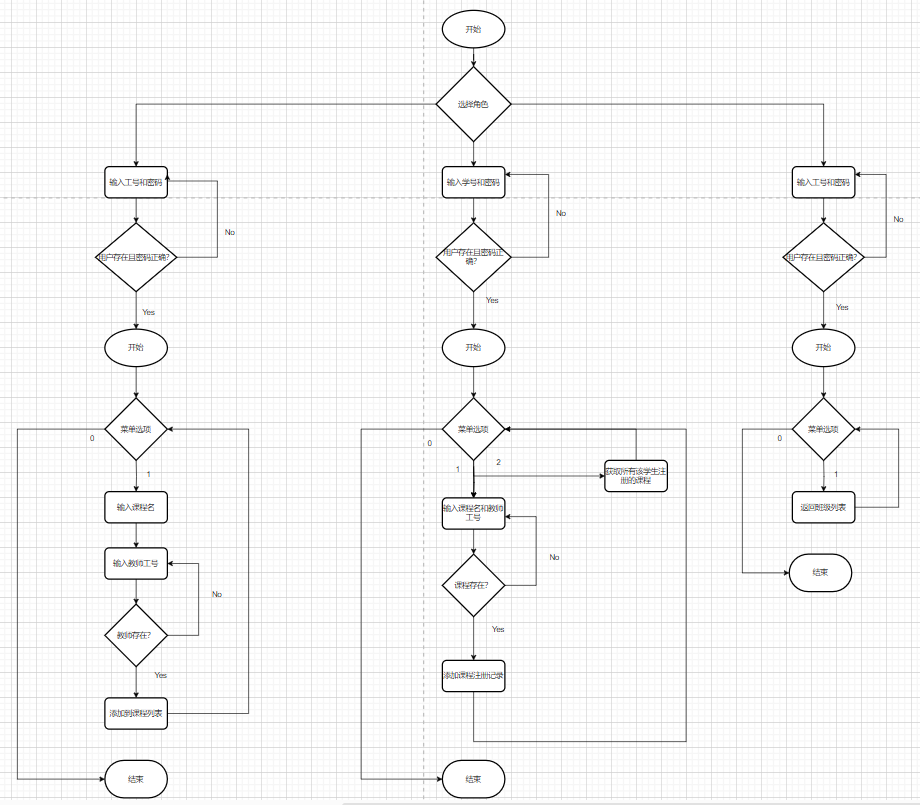
1. 课程安排模块:通过循环遍历课程注册记录列表，如果有记录的学生id与用户id一致则进行取出，最后将获取的信息循环输出。



1. 班级列表模块：通过循环遍历课程列表，取出所有教师工号等于用户工号的课程，放入新的userCourses数组，之后对userCourses数组中的每一个课新建一个Class变量cls，再循环访问注册记录列表，把所有包含该课的记录中的学生记录保存到cls的学生列表中，cls的count每次加1，搜索完成后将该变量添加到班级列表中。最后遍历班级列表进行输出。



1. 总体流程图：



* 1. 数据结构主要包括每个角色的结构体以及其数组，具体如下：

1. 角色结构体

学生结构体包括学号，姓名，密码

typedef struct{

int id;

char name[MAX\_LEN];

char password[MAX\_LEN];

}Student;

教师结构体包括工号，姓名，密码

typedef struct{

int id;

char name[MAX\_LEN];

char password[MAX\_LEN];

}Teacher;

教务处结构体包括工号，名字，密码

typedef struct{

int id;

char name[MAX\_LEN];

char password[MAX\_LEN];

}Office;

课程结构体包括课程名，所属教师

typedef struct{

char name[MAX\_LEN];

Teacher teacher;

}Course;

注册记录结构体包括课程，选课学生

typedef struct{

Course course;

Student student;

}RegistrationRecord;

班级结构体包括课程，学生列表，班级人数

typedef struct{

Course course;

int count;

Student students[MAX\_LEN];

}Class;

1. 存储数组：

用来存储所有课程

Course courses[N];

用来存储所有学生

Student students[N];

用来存储所有课程与学生之间的关系，也就是注册记录

RegistrationRecord records[N];

用来存储所有教师

Teacher teachers[N];

用来存储所有教务处人员

Office offices[N];

* 1. 外部接口通过命令行输入角色登录信息，以及学生输入注册课程信息，教务处输入课程信息；内部接口通过函数之间互相调用进行实现，比如在学生菜单中通过接收输入调用不同的函数：注册课程或查看课程安排来进行交互；对于用户界面，使用命令行来进行展示，通过键盘输入来进行交互，不同的菜单界面也会有对应的提示词，便于用户使用。
  2. 详细设计规格说明书如下：

登录模块：

输入：学号/工号，密码

处理：检查是否存在该用户

算法描述：通过循环遍历对应的角色数组，如果存在某一个记录与输入对应，则成功登录，返回该记录，否则返回学号/工号/密码错误

输出：登录成功返回该实体，并在命令行输出提示，登录失败输出重新输入提示。

新增课程模块：

输入：课程名，教师工号

处理：检查是否存在该用户，若存在则将该课程添加到课程列表。

算法描述：通过循环遍历教师列表，如果存在某一个记录的工号与输入对应，则将新的课程添加到课程列表中，否则提示错误。

输出：若添加成功，则返回提示并将新的课程添加到列表中，若登录失败给出提示。

学生注册模块：

输入：课程名，教师工号

处理：检查是否存在该课程，若存在则将该注册记录追加到记录列表。

算法描述：通过循环遍历课程列表，如果存在某一个记录的课程名与教师工号与输入对应，则将用户与该课程合成一条记录追加到记录列表，否则提示错误。

输出：若添加成功，则返回提示并将新的记录追加到列表中，否则给出提示。

课程安排模块:

输入：无

处理：搜索所有包含用户的课程注册记录，并将信息输出

算法描述：通过循环遍历课程注册记录列表，如果有记录的学生id与用户id一致则进行取出，最后将获取的信息循环输出。

输出：所有该学生的课程信息，包括课程名，教师。

班级列表模块:

输入：无

处理：取出所有用户执教的课，再从注册记录中取出所有注册该课的学生信息，用Class数组激励。

算法描述：通过循环遍历课程列表，取出所有教师工号等于用户工号的课程，放入新的userCourses数组，之后对userCourses数组中的每一个课新建一个Class变量cls，再循环访问注册记录列表，把所有包含该课的记录中的学生记录保存到cls的学生列表中，cls的count每次加1，搜索完成后将该变量添加到班级列表中。最后遍历班级列表进行输出。

输出：所有的班级信息，包括课程名，人数，所有选课学生的姓名等信息。

（3）完整代码实现如下：

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

const int N = 1e5+10;

const int MAX\_LEN = 100;

typedef struct{

int id;

char name[MAX\_LEN];

char password[MAX\_LEN];

}Student;

typedef struct{

int id;

char name[MAX\_LEN];

char password[MAX\_LEN];

}Teacher;

typedef struct{

int id;

char name[MAX\_LEN];

char password[MAX\_LEN];

}Office;

typedef struct{

char name[MAX\_LEN];

Teacher teacher;

}Course;

typedef struct{

Course course;

Student student;

}RegistrationRecord;

typedef struct{

Course course;

int count;

Student students[MAX\_LEN];

}Class;

int studentIndex=0;

int courseIndex=0;

int teacherIndex=0;

int officeIndex=0;

int recordIndex=0;

Course courses[N];

Student students[N];

RegistrationRecord records[N];

Teacher teachers[N];

Office offices[N];

Student curStudent;

Teacher curTeacher;

void addTeacher(int id,char\* name,char\* password){

Teacher teacher;

teacher.id=id;

strcpy(teacher.name,name);

strcpy(teacher.password,password);

teachers[teacherIndex++]=teacher;

}

void addStudent(int id,char\* name,char\* password){

Student student;

student.id=id;

strcpy(student.name,name);

strcpy(student.password,password);

students[studentIndex++]=student;

}

void addOffice(int id,char\* name,char\* password){

Office office;

office.id=id;

strcpy(office.name,name);

strcpy(office.password,password);

offices[officeIndex++]=office;

}

void addCourse(){

Course newCourse;

printf("--------添加课程-------\n");

printf("请输入课程名：\n");

scanf("%s",newCourse.name);

printf("请输入任课教师工号：\n");

int id;

int flag=0;

do{

scanf("%d",&id);

for(int i=0;i<teacherIndex;i++){

if(teachers[i].id==id){

flag=1;

newCourse.teacher=teachers[i];

courses[courseIndex++]=newCourse;

break;

}

}

if(!flag){

printf("找不到该教师\n");

}

}while(!flag);

printf("添加成功！\n");

}

void registerCourse(){

int flag=0;

do{

char\* cname;

int id;

printf("请输入课程名称\n");

fflush(stdin);

scanf("%s",cname);

fflush(stdin);

printf("请输入老师工号\n");

scanf("%d",&id);

for(int i=0;i<courseIndex;i++){

if(!strcmp(courses[i].name,cname)&&courses[i].teacher.id==id){

RegistrationRecord newRecord;

newRecord.course=courses[i];

newRecord.student=curStudent;

records[recordIndex++]=newRecord;

flag=1;

break;

}

}

if(!flag){

printf("未找到课程\n");

}

}while(!flag);

printf("选课成功\n");

}

void showAllCourses(){

int len=0;

Course userCourses[MAX\_LEN];

for(int i=0;i<recordIndex;i++){

if(records[i].student.id==curStudent.id){

userCourses[len++]=records[i].course;

}

}

printf("您的课程列表为:\n");

for(int i=0;i<len;i++){

printf("课程名：%s,教师：%s\n",userCourses[i].name,userCourses[i].teacher.name);

}

}

void showClasses(){

int len=0;

Course userCourses[MAX\_LEN];

for(int i=0;i<courseIndex;i++){

if(courses[i].teacher.id==curTeacher.id){

userCourses[len++]=courses[i];

}

}

int len2=0;

Class\* userClasses = (Class\*)malloc(MAX\_LEN \* sizeof(Class)); // Dynamically allocate memory for userCourses

for(int i=0;i<len;i++){

Class cls;

cls.count=0;

for(int j=0;j<recordIndex;j++){

if(!strcmp(records[j].course.name,userCourses[i].name)&&(records[j].course.teacher.id==curTeacher.id)){

cls.students[cls.count++]=records[j].student;

}

}

cls.course=userCourses[i];

userClasses[len2++]=cls;

}

printf("-------您的班级列表-------\n");

for(int i=0;i<len2;i++){

printf("%s\n",userClasses[i].course.name);

printf("总人数: %d\n",userClasses[i].count);

printf("\*\*\*\*学生列表\*\*\*\*\n");

for(int j=0;j<userClasses[i].count;j++){

printf("学号：%d, ",userClasses[i].students[j].id);

printf("姓名：%s\n",userClasses[i].students[j].name);

}

}

}

void studentMenu(){

int input;

while(1){

printf("-------学生界面-------\n");

printf("1. 注册课程\n");

printf("2. 查看课程安排\n");

printf("0. 退出\n");

scanf("%d",&input);

switch(input){

case 0:

return;

break;

case 1:

registerCourse();

break;

case 2:

showAllCourses();

break;

default:

printf("invalid choice\n");

}

}

}

void officeMenu(){

int input;

while(1){

printf("------教务处界面------\n");

printf("1. 新增课程信息\n");

printf("0. 退出\n");

scanf("%d",&input);

switch(input){

case 0:

return;

break;

case 1:

addCourse();

break;

default:

printf("invalid choice\n");

}

}

}

void teacherMenu(){

int input;

while(1){

printf("------教师界面------\n");

printf("1. 查看班级列表\n");

printf("0. 退出\n");

scanf("%d",&input);

switch(input){

case 0:

return;

break;

case 1:

showClasses();

break;

default:

printf("invalid choice\n");

}

}

}

void loginAsOffice(){

printf("------教务处登录------\n");

int flag=0;

do{

int id;

char\* password;

printf("请输入id\n");

scanf("%d",&id);

printf("请输入密码\n");

fflush(stdin);

scanf("%s",password);

fflush(stdin);

for(int i=0;i<officeIndex;i++){

if(offices[i].id==id&&!strcmp(offices[i].password,password)){

flag=1;

}

}

if(!flag){

printf("密码或用户名错误！\n");

}

}while(!flag);

printf("-----login successful!-----\n");

officeMenu();

}

void loginAsStudent(){

printf("------学生登录------\n");

int flag=0;

do{

int id;

char\* password;

printf("请输入id\n");

scanf("%d",&id);

printf("请输入密码\n");

fflush(stdin);

scanf("%s",password);

fflush(stdin);

for(int i=0;i<studentIndex;i++){

if(students[i].id==id&&!strcmp(students[i].password,password)){

flag=1;

curStudent=students[i];

}

}

if(!flag){

printf("密码或用户名错误！\n");

}

}while(!flag);

printf("-----login successful!-----\n");

studentMenu();

}

void loginAsTeacher(){

printf("------教师登录------\n");

int flag=0;

do{

int id;

char\* password;

printf("请输入id\n");

scanf("%d",&id);

printf("请输入密码\n");

scanf("%s",password);

fflush(stdin);

for(int i=0;i<teacherIndex;i++){

if(teachers[i].id==id&&!strcmp(teachers[i].password,password)){

curTeacher=teachers[i];

flag=1;

}

}

if(!flag){

printf("密码或用户名错误！\n");

}

}while(!flag);

printf("-----login successful!-----\n");

teacherMenu();

}

void prework(){

addOffice(1,"admin","123456");

addTeacher(101,"lisi","ls123");

addStudent(201,"zhangsan","zs123");

addStudent(202,"xiaoming","xm123");

addStudent(203,"lili","ll123");

Course course;

strcpy(course.name,"english");

course.teacher=teachers[0];

courses[courseIndex++]=course;

}

int main(){

prework();

while(1){

int input;

while (1){

printf("-------欢迎来到课程管理系统-------\n");

printf("1. 教务处登录\n");

printf("2. 学生登录\n");

printf("3. 教师登录\n");

printf("0. 退出\n");

scanf("%d",&input);

switch(input){

case 0:

return 0;

case 1:

loginAsOffice();

break;

case 2:

loginAsStudent();

break;

case 3:

loginAsTeacher();

break;

default:

printf("invalid choice\n");

}

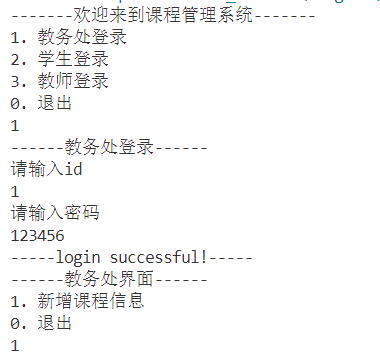
}

}

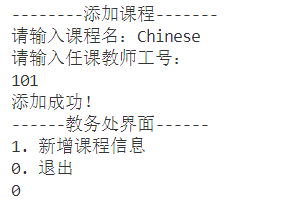
}

测试截图：

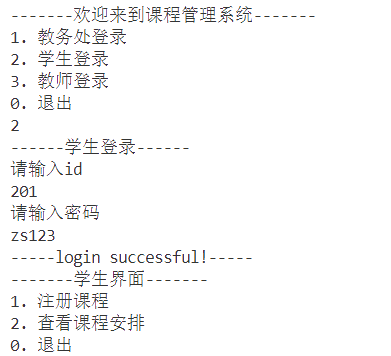
教务处登录：



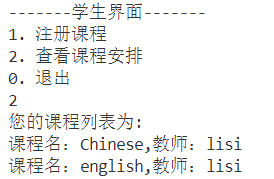
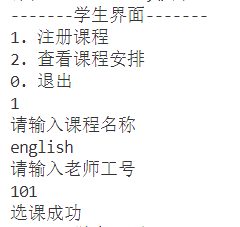
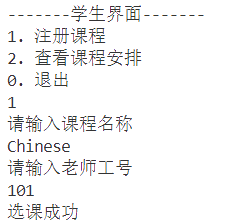
教务处新增课程信息：



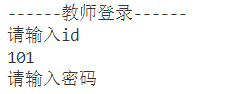
学生登录：

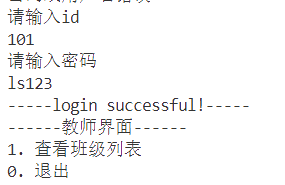


学生注册课程与查看课程安排：



教师登录：





查看班级列表：

