

C语言课程设计——公共交通行程管理系统

一、前言

1. 编写背景：

我国城市交通经历了从改革开放初期以自行车为主要交通工具，到20世纪90年代以公共汽电车为主体的发展阶段，在21世纪初，国家提出了公共交通优先发展的战略，城市交通进入了统筹协调、综合发展的新阶段。城市客运量在改革开放初期仅有132亿人次，到2018年已增至1262亿人次。近年来，城市公共汽电车车辆数与运营里程持续增长，轨道交通高速发展，传统出租数量稳中略降，互联网出行新业态涌现。城市交通呈现多元化发展。城市公共汽电车车辆数和运营线路长度持续大幅增长，截至2018年底，我国城市公共汽电车运营车辆数达67.34万辆，运营线路达60590条，运营线路总长度达119.95万公里。城市轨道交通自1969年北京市开通全国第一条城市轨道交通线路以来，截至2018年底，已有35个城市开通运营城市轨道交通，运营线路达到171条，2014-2018年，公共汽电车与出租汽车客运量逐年下降，轨道交通运输量却在逐年上升，轨道交通在城市客运中的地位越来越重要；但从绝对数来看，公共汽电车客运量占客运总量的比例仍很高，公共汽电车仍是城市客运的主要工具出租汽车客运量略有下降，但总体较稳定。运营里程达5295.1公里，完成客运量212.77亿人次，运营里程及完成客运量均居世界第一。其次，随着信息技术的发展，公交智能化深入推进，移动支付、智能公交电子站牌有效提升了顾客的乘车体验。

2020新冠疫情突然爆发，全国全世界都受到巨大的冲击与影响，就业、医疗、公共基础设施等面临的问题接踵而来，公共交通出行更是面临着巨大的问题与挑战。疫情之下严控人流密集场所，公交地铁受冷落。交通运输部颁布《客运场站和交通运输工具新冠肺炎疫情分区分级防控指南》，严控高风险地区的公交车拥挤度在每平方米 ≤ 4 人，地铁满载率、拥挤度不能超过70%。恒大研究院统计得出，上海、广州等7城地铁运量在疫情期间大幅下降，2020年春节第一周客运总量仅为2019年同期的8.8%，第四周（22-25日）有所上升，但仍仅22.3%。

伴随中国新型冠状病毒肺炎(以下简称“新冠肺炎”)疫情防控向好态势进一步巩固，防控工作任务正在转向疫情防控与恢复经济社会运行两手抓，防控要求也从静态隔离转向动态防控。公共交通的出行特点，决定了其将在经济社会秩序恢复过程中担当疫情防控“主战场”角色。如果以一日出行链来还原城市中普通人的主要活动，可以发现，在家庭、办公或社交场所以及交通工具三类空间中，公共交通载运工具的传播风险相对较高。可以预见，伴随经济活动的活跃度逐步提升，素有城市血脉之称的交通系统将迎来真正意义上的持续考验。其中，公共交通的角色可能面临大幅转型，走出严格防控模式下的低活跃度状态，承担疫情防控“主战场”和经济恢复“保障线”的双重角色。

在此背景下，疫情防控与公共交通出行信息紧密结合，成为广大人民群众日常生活中不可或缺的重要安全保障，因此，编者编写了这个公共交通乘客行程系统，旨在模拟疫情管控与公共交通安全的结合，学习掌握C语言相关知识的同时深入了解公共交通在疫情爆发后的现状与未来的发展趋势，在提升自我专业能力的同时拓宽视野，增长知识。

2. 编写目的：

通过对公共交通出行服务类网站及app的使用者的调查和分析，对铁路12306及通信行程卡等应用的使用，对一些技术指标的学习和网络资料查询，我们编写出这一份终期报告。

本报告对于整个“公共交通乘客行程系统”进行了全面的用户需求和功能分析。包括可行性分析，需求分析，系统功能设计，代码实现，软件亮点和不足，集成测试等等。本报告明确了本软件系统架构设计，软件结构与数据结构设计，各模块之间的接口和调用，系统界面设计，系统功能设计，具体算法设计以及整个软件的源代码。

同时，该项报告也对代码进行一定程度的解释与概括，增强了后期测试人员对于软件的调试和验收的可读性与可修改性。

本报告的预期受众为全年龄群体包括所有有公共交通出行需求的用户。

3. 参考资料：

1. 王士元. C高级实用程序设计. 北京: 清华大学出版社. 1996年。
2. 周纯杰, 何顶新等. 程序设计与应用 (用C/C++编程). 北京: 机械工业出版社. 2008年。

4. 参考软件：

铁路12306、通信行程卡等

二、任务概述

1. 目标功能

本公共交通乘客行程模拟系统可以实现同类app、网站的大多基础功能，在欢迎界面进行登陆类型的选择或者注册。

1. 用户进入后先进行注册或登录，登陆后进行身份证信息与电话号码的绑定，绑定完成后可以进行个人健康状况、身份信息等个人信息的查看，可以进行乘车前的登记，查询个人车次并查询行程及途径站点，还可以对个人的乘车记录进行查询，搜索自己乘坐过的列车及到达过的城市。
2. 管理员登陆后可以对疫情状况的调查与管控，管理员可以模拟输入乘客信息进行对乘车人员身体健康状况的标记，还可以查询已经进行标记的阳性乘客，查询阳性乘客乘坐的列车，并查询与阳性乘客有相关接触的密接乘客。

2. 编写规范

1. 命名规范

函数的命名一律以小写英文为规范，文件与变量的命名实现英文释义与其功能的对应，数据结构等的命名以上述参考资料为标准进行规范化编程。

2. 注释

- a. 函数功能在函数原型后注明。
- b. 难以理解的算法和流程给出相应的注释。

3. 内存管理

- a. 保证对指针的malloc和free一一对应
- b. 保证文件的fopen和fclose一一对应

4. 页面控制函数编写规范

三、运行环境和配置

1. 硬件接口

处理器：Intel Pentium 166 MX 或以上

硬盘：空间150MB以上。

屏幕适配器：VGA接口。

系统运行内存：要求32MB以上。

2. 软件接口

开发软件工具：Borland C++ 3.1。

文字编辑工具：Notepad++、Borland C++ 3.1。

数据库：文本存储（记事本）。

操作系统：DOS WINDOWS 9X/ME/2000/XP/WINDOWS 10等。

3. 控制

该系统通过鼠标与键盘直接进行控制。用户将鼠标移至需要操作的功能区进行点击，同时通过键盘来完成登陆、注册、搜索、填写信息等输入功能。点击返回箭头即可返回上一页面。通过中断技术来获取鼠标的位置与键盘的输入功能。

四、需求分析

公共交通乘客行程模拟系统主要为了实现出行记录查询功能，出行登记功能，乘客健康状况登记功能，各种信息查询和浏览功能。具体包括：

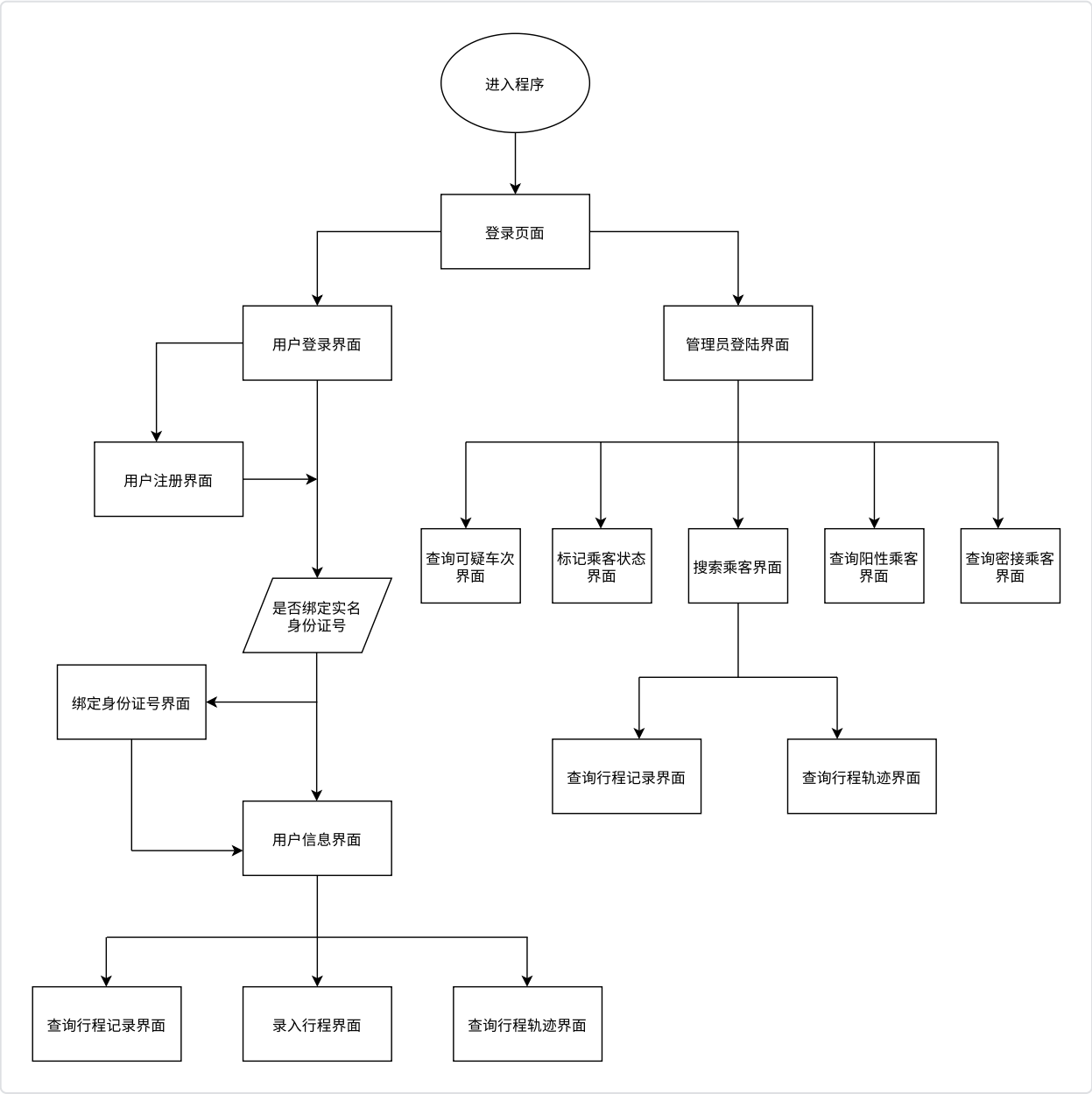
1. 为保证疫情期间乘客的个人健康安全，每个乘客都可以查看乘客个人信息，查询个人健康状况的更新，公共交通现面临严峻的防疫安全问题，乘客需在出行前确认自身身体健康状况以免对公众安全带来风险，以保证每一位使用公共交通的乘客的安全问题，健康状况处于非正常的乘客将被暂时拒绝使用公共交通设施以免为广大乘客带来传染风险。
2. 为保证每一位使用公共交通出行的乘客的健康状况，确认使用公共交通进行出行的乘客的安全状况，乘客需要在上车前进行乘车登记。乘客应提前了解个人乘坐列车的起点站终点站与途径站点并了解列车出行日期，乘客希望避免出行过程出现错误或者面临与感染者同乘而被感染的风险，因此乘客需提前确认好个人所乘列车的乘车时间和列车班次，并提前确认起点、目的地与途径站点。确认完成后凭借登记乘车凭证才可登上列车，未进行乘车登记或者乘车登记未通过的乘客都无法登车。用户希望对个人乘车记录进行查看以确认自身是否同感染者有过时空交集以对自身健康状况进行确认。同时，用户通过查看列车途径路线以确定行车途中乘客经过的地区方便进一步对自身健康状况进行确认。
3. 管理员需要对乘客的信息进行检查和管理时，通过管理员权限对乘客的身体健康状况进行录入，同时为了管理员更方便的对患病的乘客进行管理，我们设计了查询查询全体阳性乘客的功能，以便于政府机构、医院等及时搜索并获取信息对阳性感染者进行控制与隔离，将疫情期间的防疫措施落实到位。同时，如果能查询阳性患者乘坐过的车辆，将会进一步便利管理员与政府机关等对疫情防控的管理，所以我们加入了对可疑车次的查询，以便搜索乘坐过相同车次的乘客并对其健康状况进行管理并排查。并将与阳性乘客有相关接触的乘客采集，进行对密接乘客的管理与排查，方便进行下一步的疫情管控措施。

本系统需要完成的具体任务如下：

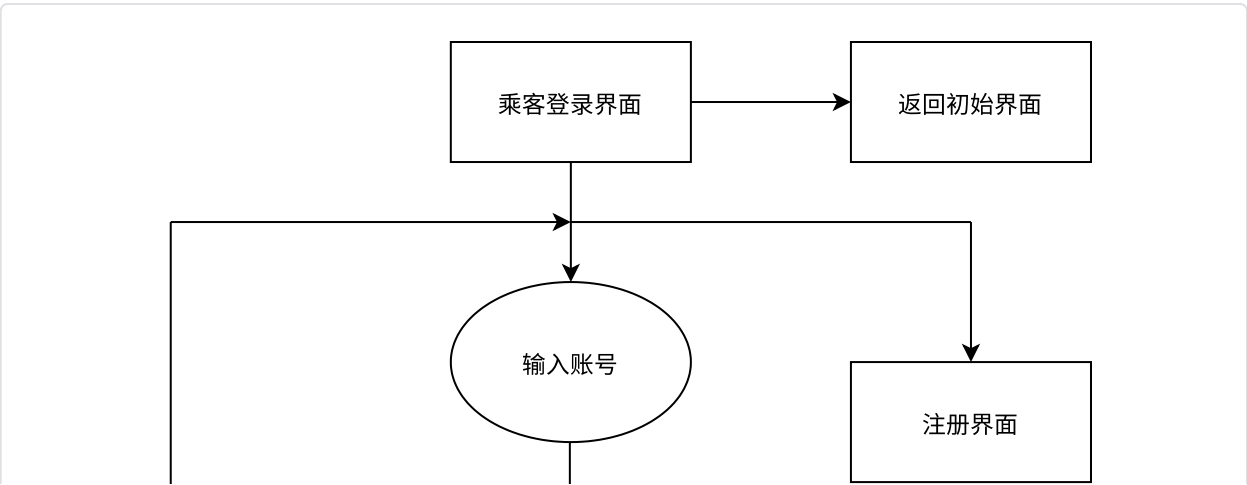
1. 登录注册。用户通过注册获得与身份证唯一绑定的账号，并可登录该账号进行操作。每个用户的用户信息、乘车信息等都和账号绑定。
2. 乘车登记功能。用户可在此界面选择称作班次，确认乘车时间及路线，在登车前完成登记,用户输入列车车次后可以选择起点站与终点站，将乘客起点站与终点站录入已完成乘车前的乘车具体登记。
3. 乘车记录查询功能。用户可在此界面查询个人乘坐的车次及路径，系统将展示乘客乘坐车次及乘坐车次的起点站、终点站。
4. 轨迹查询功能。用户可以在此界面查看个人乘坐列车的途径路线，并查看路线上的途径站点。
5. 管理员录入功能。管理员输入乘客身份证号确认信息后选择乘客健康状况对乘客的身体家康状况录入系统，方便乘客及管理员等进行查看。
6. 阳性乘客查询功能。根据已录入的信息和乘客的健康状况，对列车阳性乘客进行统一查询，其中可查看乘客的详细信息如感染症状、姓名、身份证号、电话、性别等与个人信息绑定的重要信息。
7. 可疑列车查询功能。系统判定阳性患者所乘坐过的列车为可疑列车，系统通过查询阳性乘客的乘车记录以确定可能带有病毒的列车。查询的结果包括列车名称，列车路线及铁路干线途经城市。
8. 密接乘客查询功能。管理员在此界面查询与阳性患者乘坐同一列车的乘客的个人身份信息，包括乘客的身份证号，电话，姓名等个人信息。
9. 搜索功能。

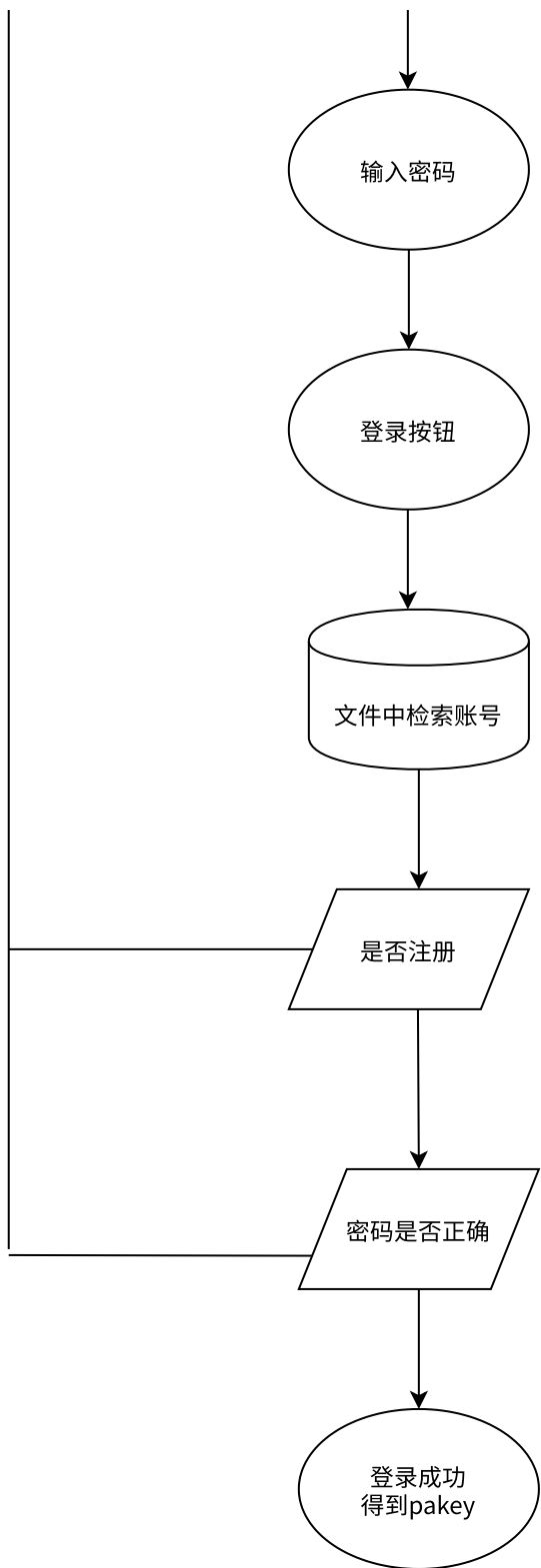
五、系统设计

1. 主要框架

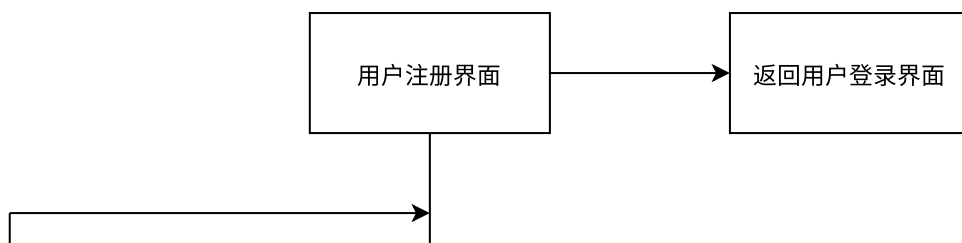


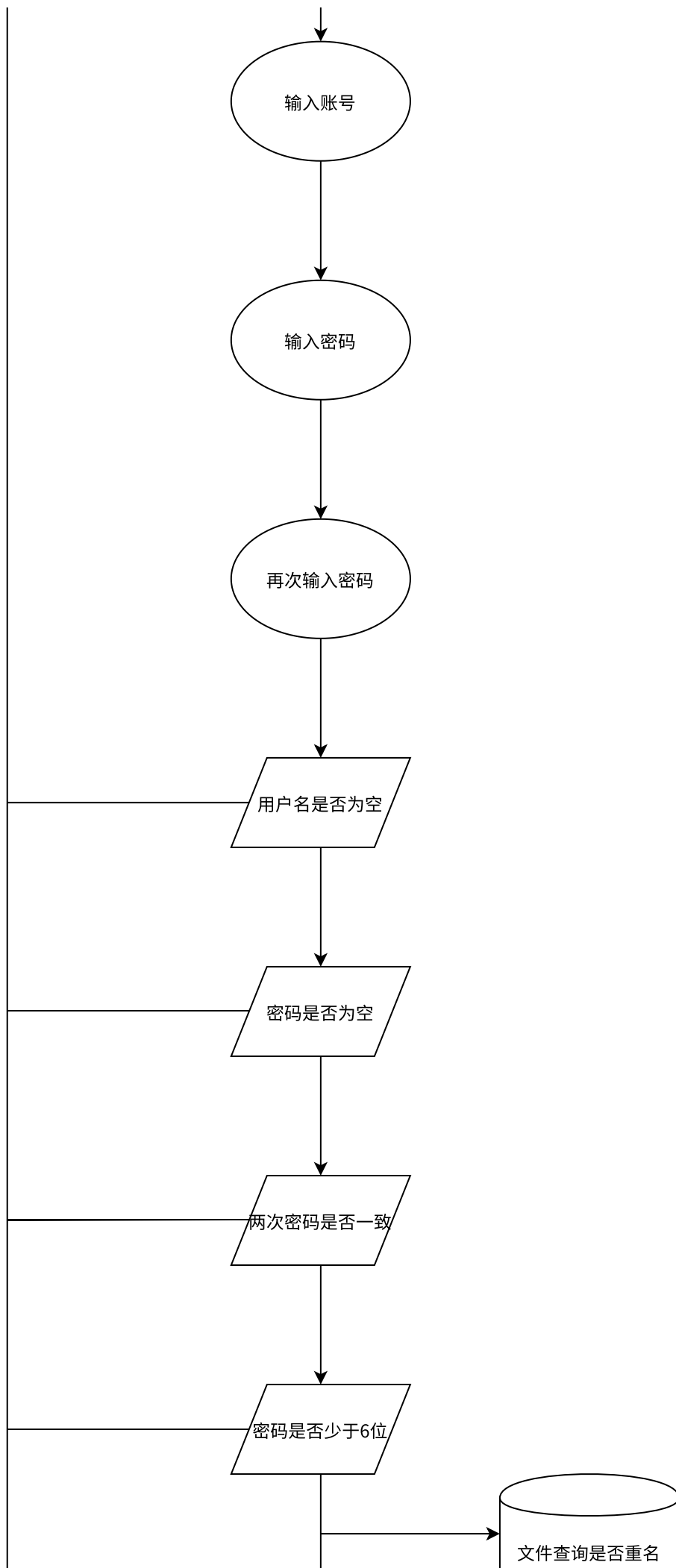
2. 用户登录

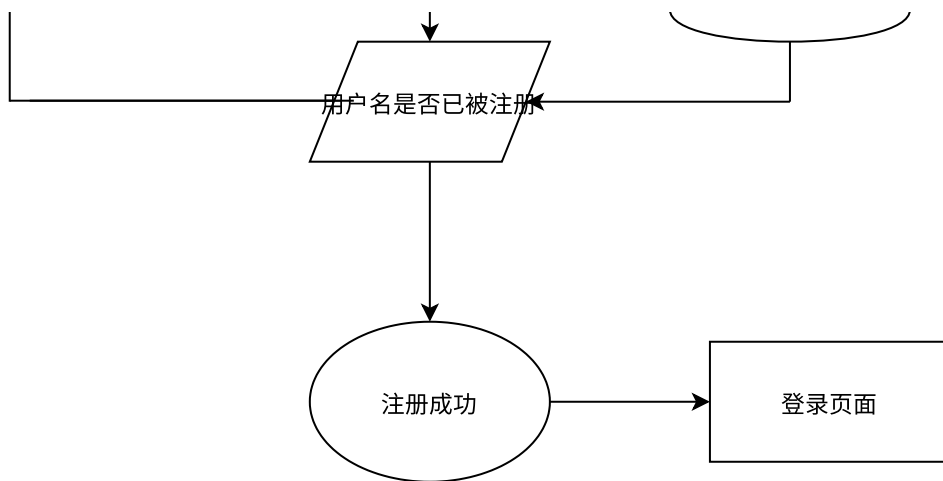




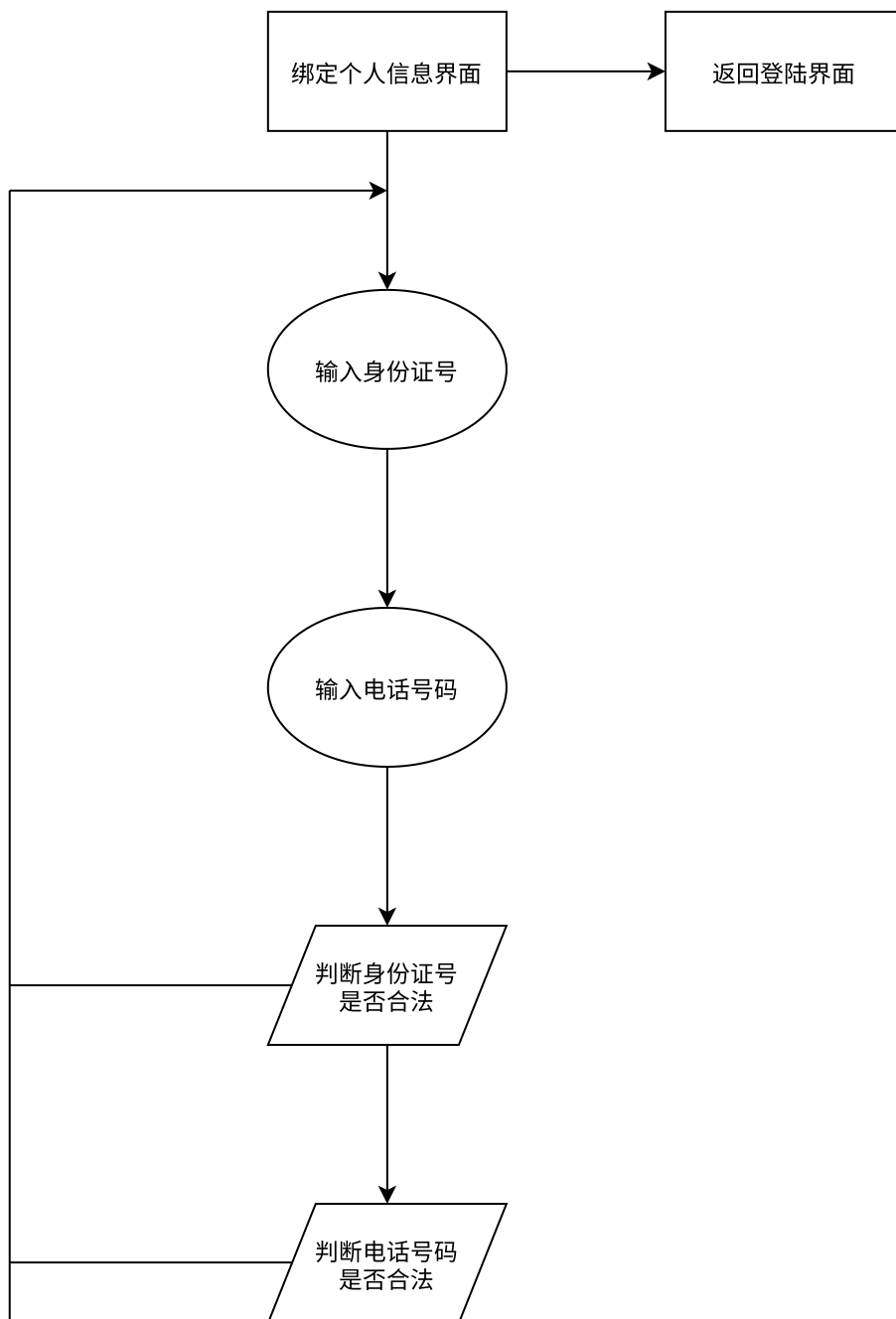
3. 用户注册

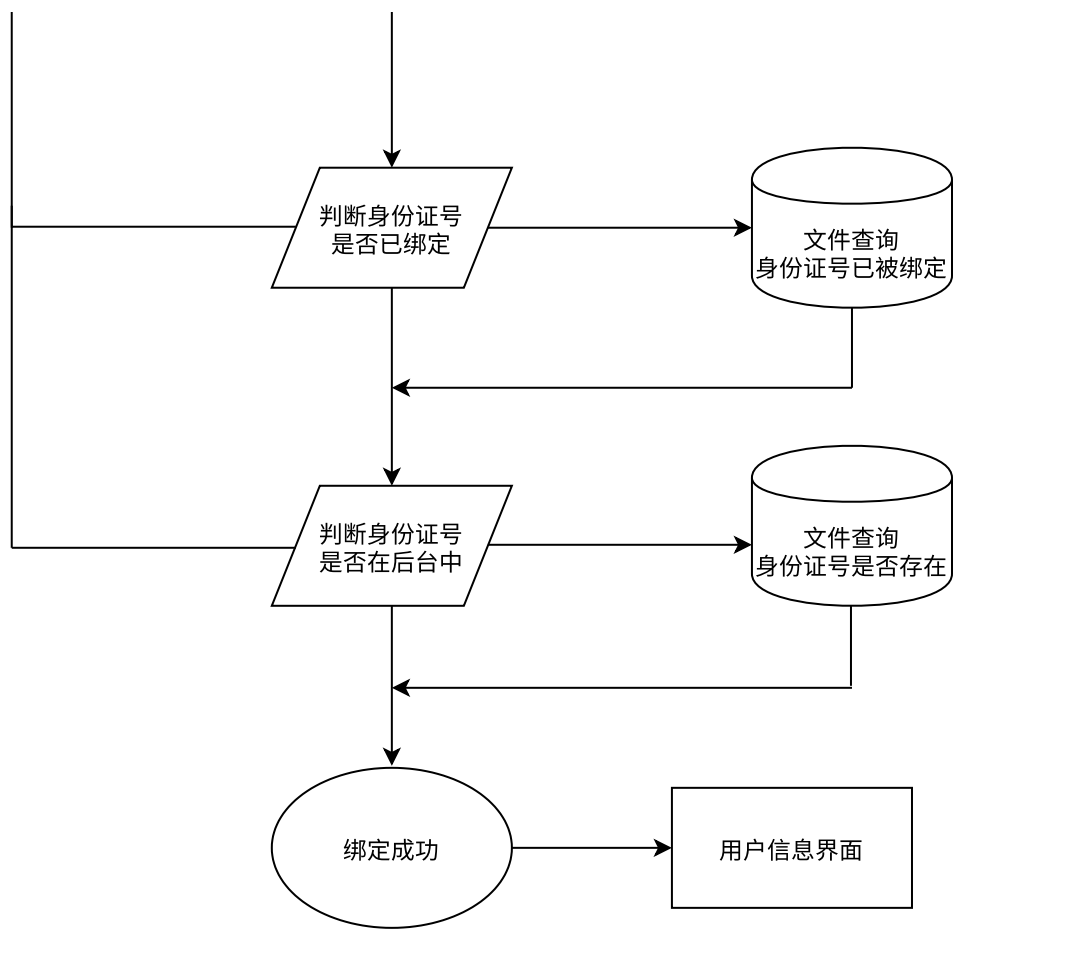




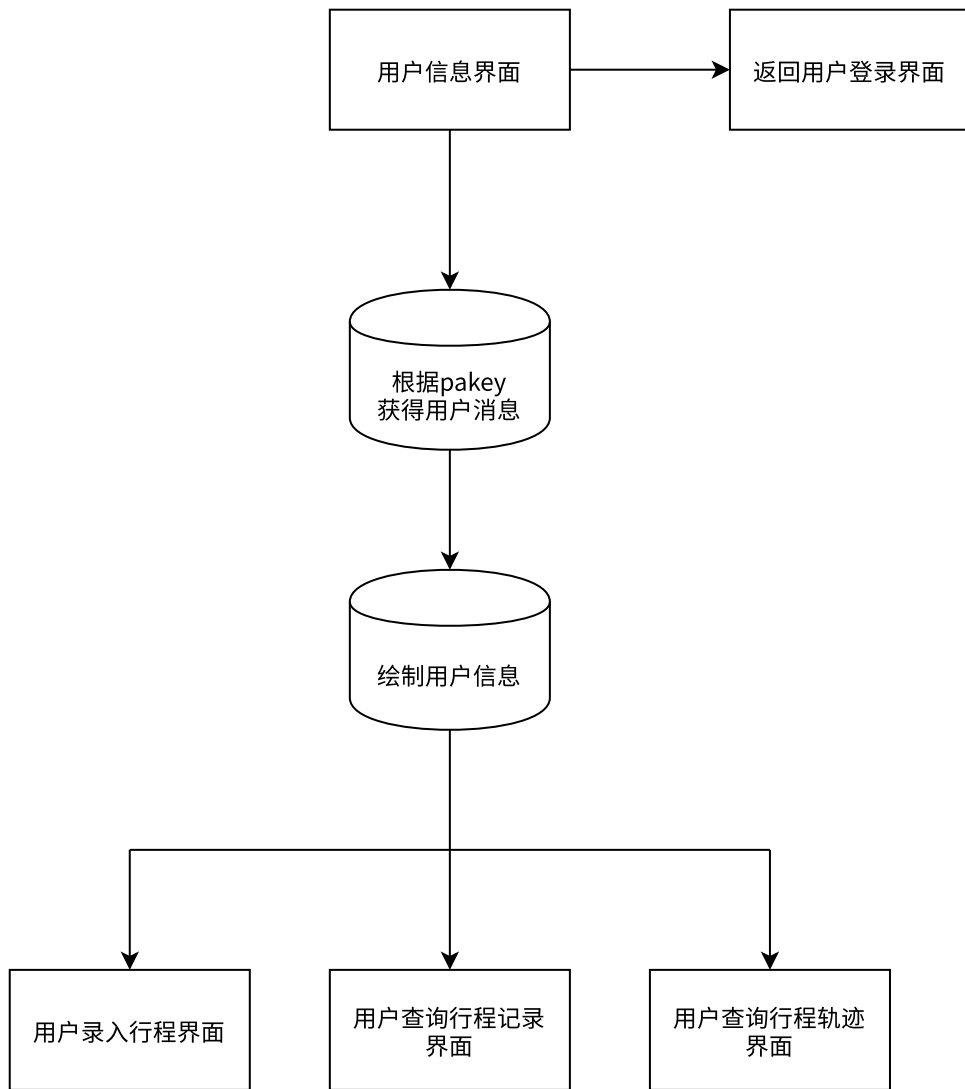


4. 用户绑定身份信息

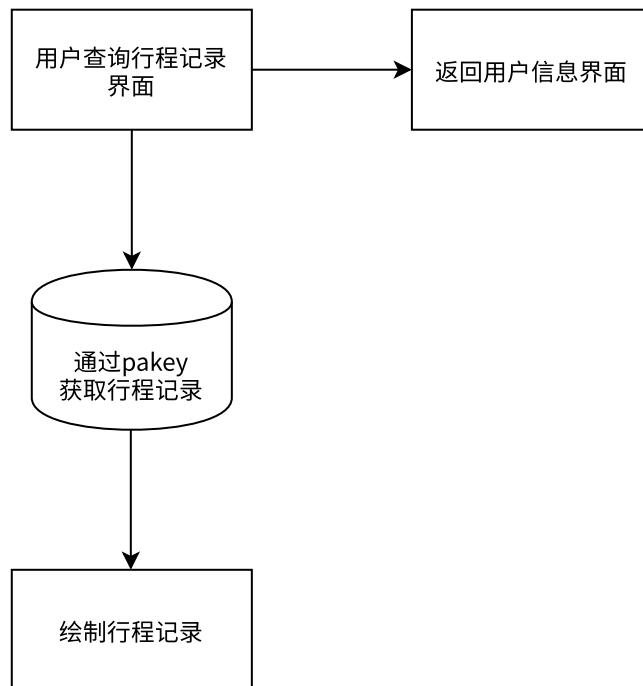




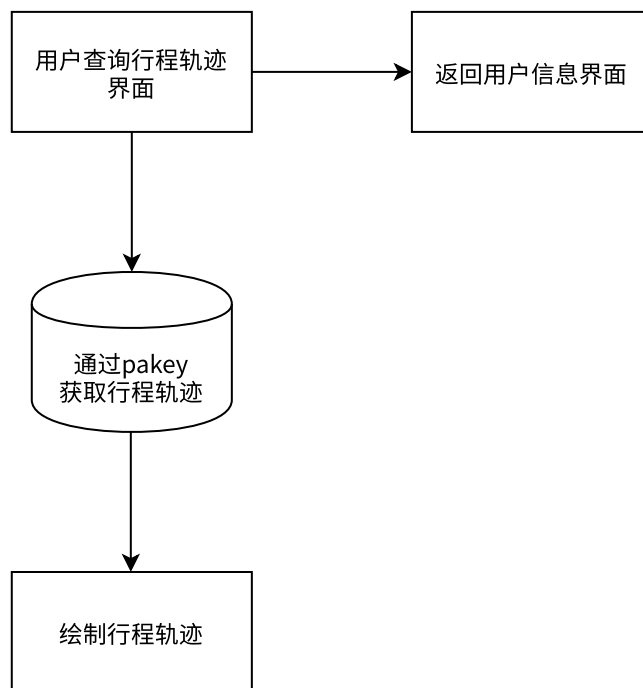
5. 用户信息界面



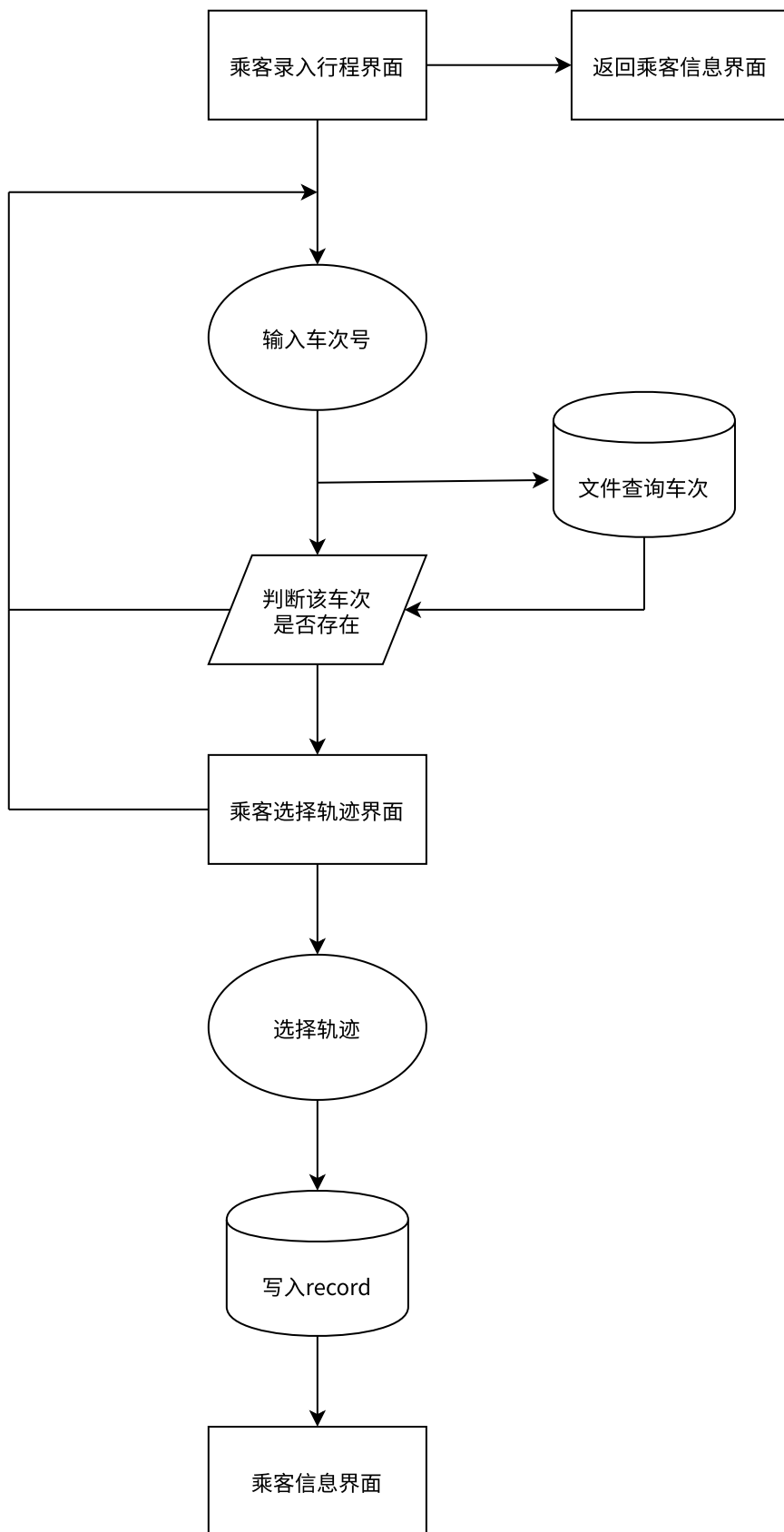
6. 用户查询行程记录



7. 用户查询行程轨迹

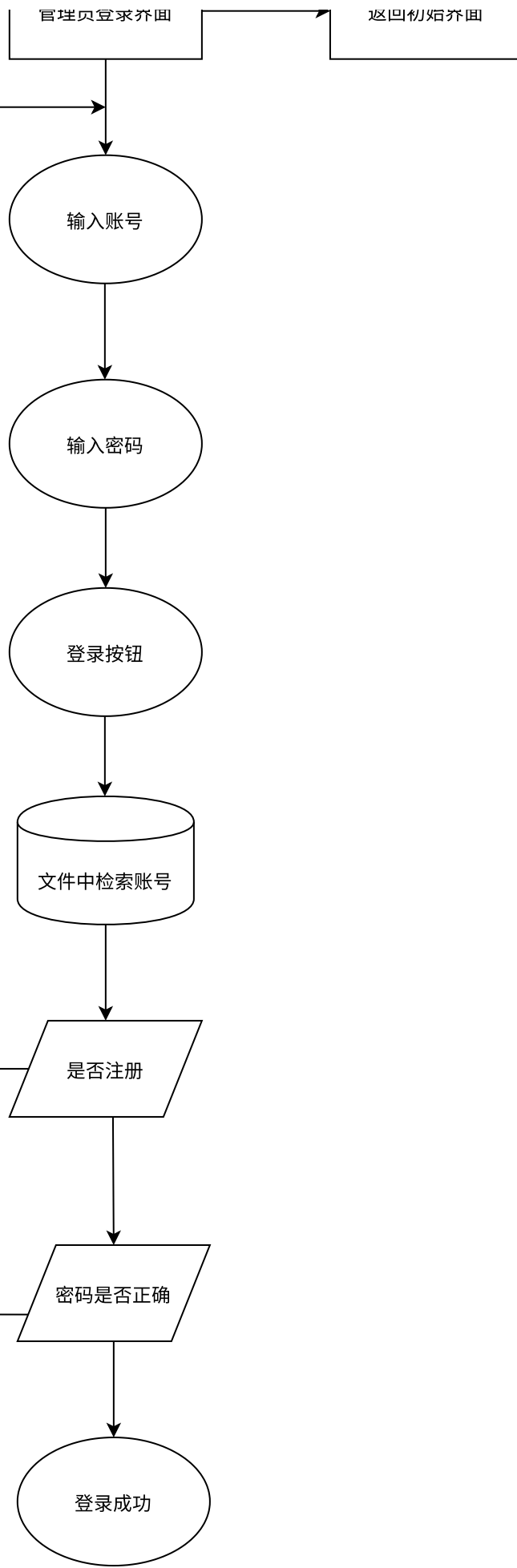


8. 用户录入行程

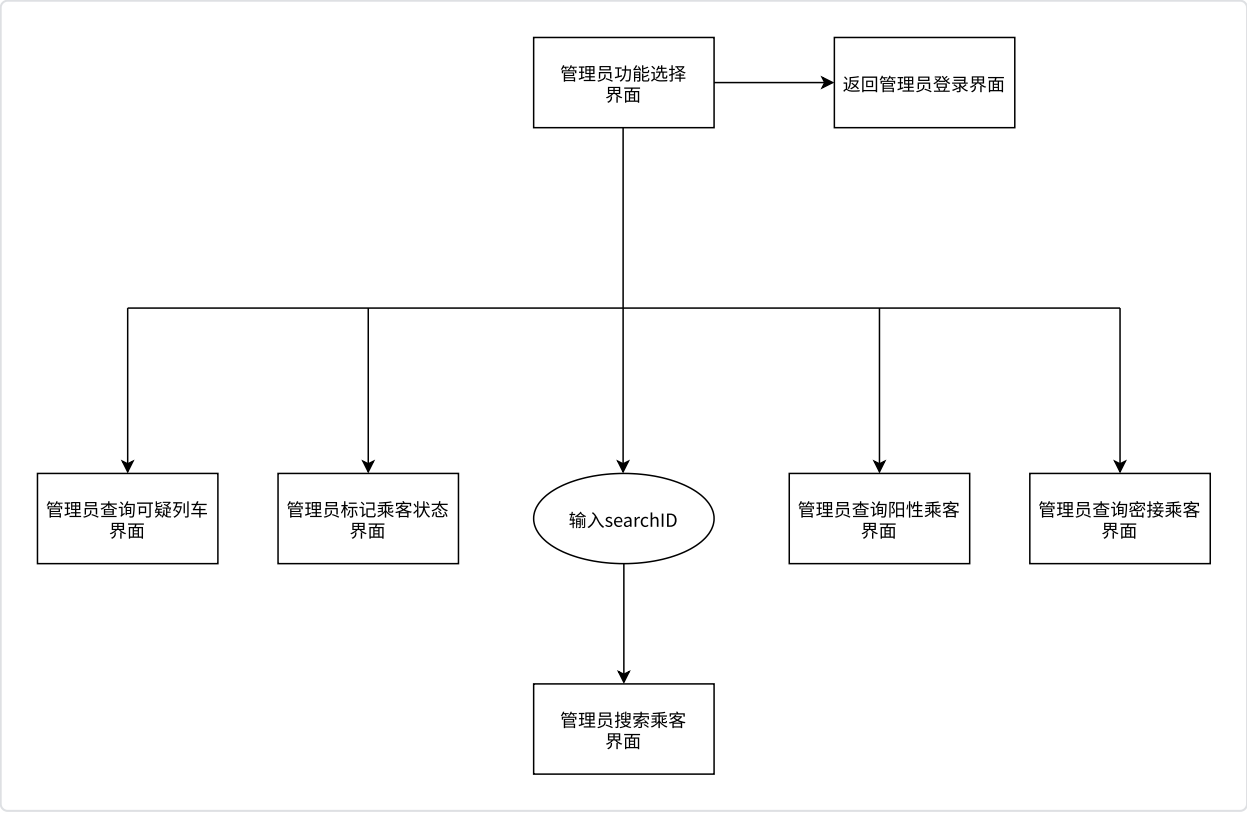


9. 管理员登录

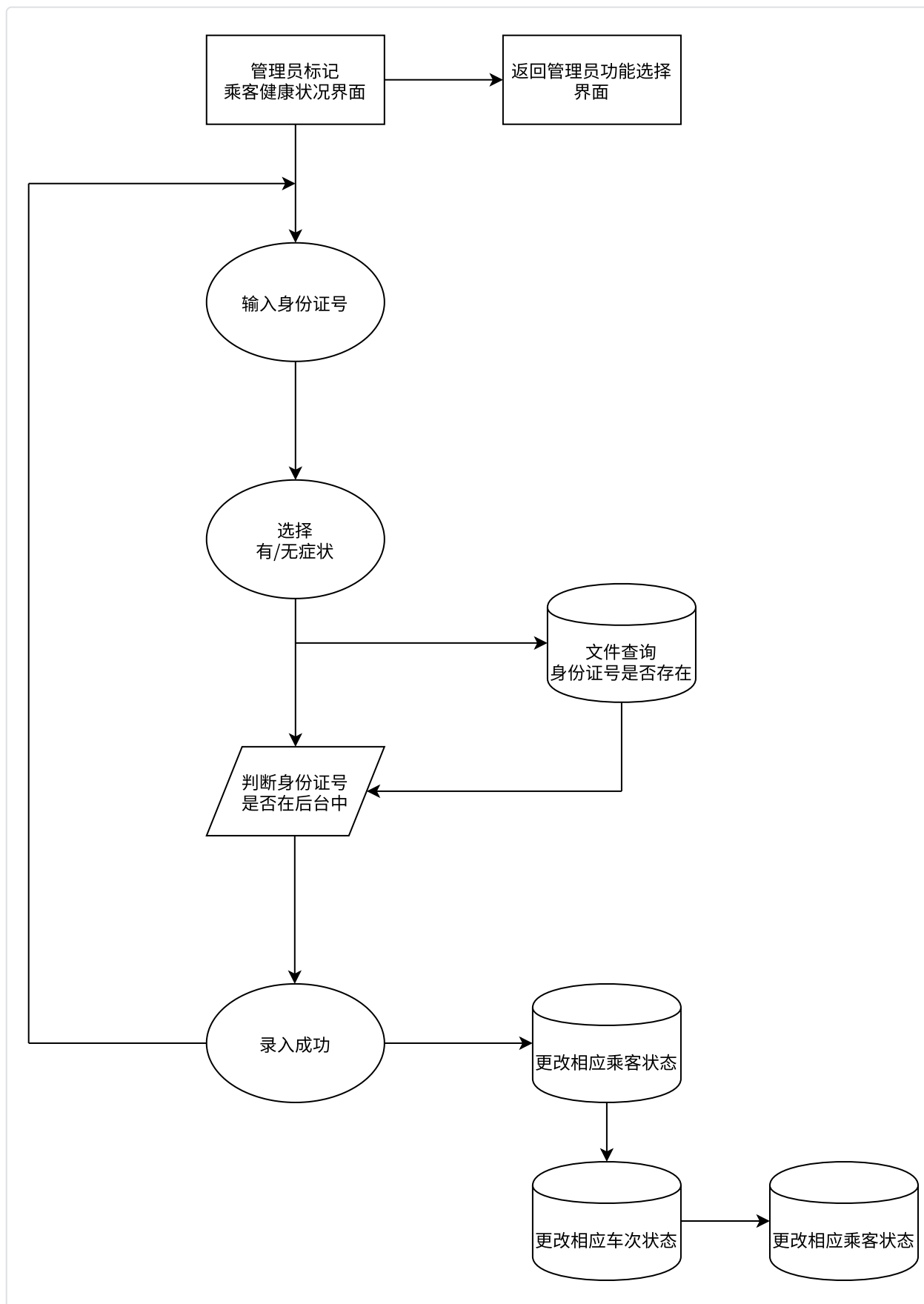




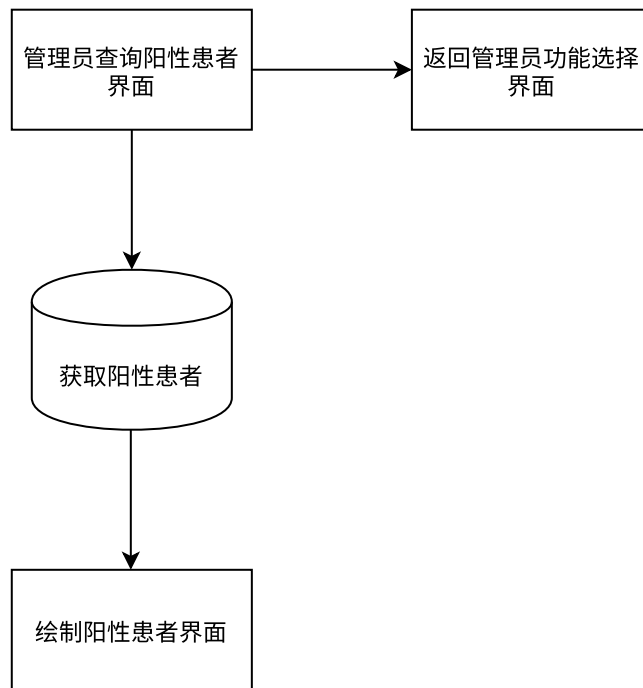
10. 管理员功能选择界面



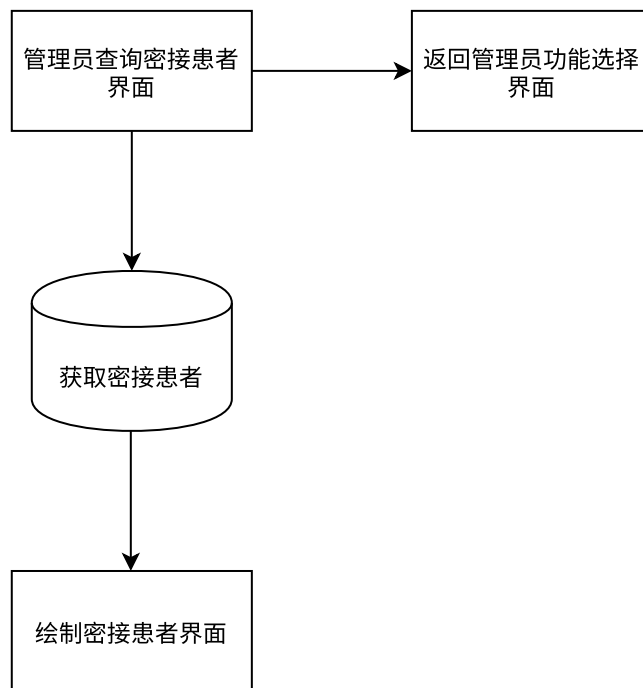
11. 管理员标记乘客健康状况



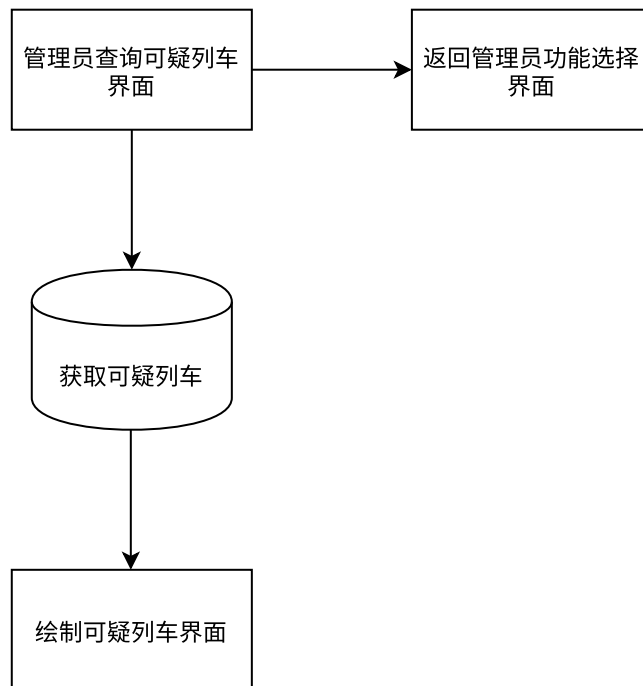
12. 管理员查询阳性患者



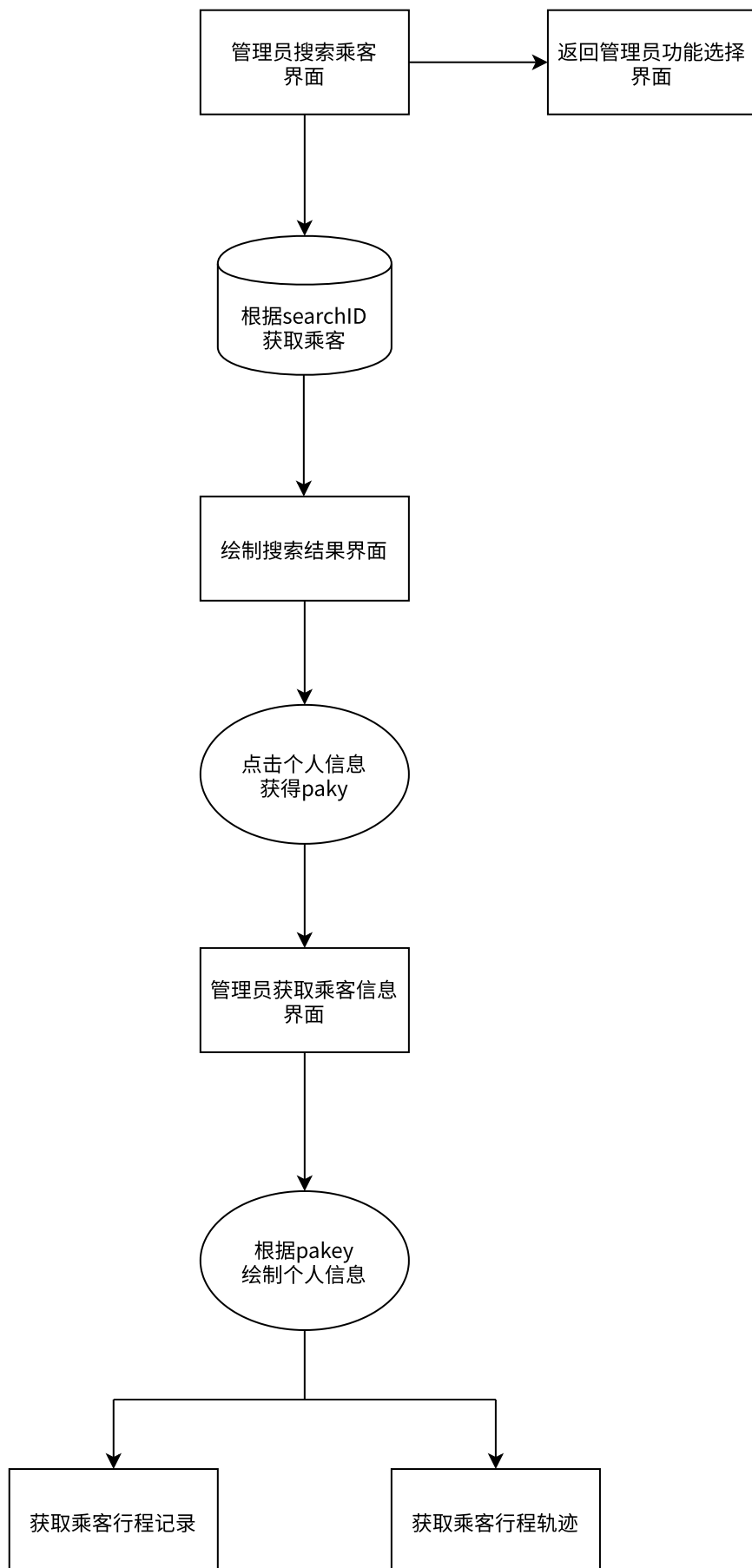
13. 管理员查询密接患者



14. 管理员查询可疑列车



15. 管理员搜索乘客



六、界面设计

七、数据结构设计

本程序共定义了6种结构体，声明于data.h中

```
C

1  #ifndef _DATA_H_
2  #define _DATA_H_
3
4  typedef struct administrator
5  {
6      int adkey;
7      char name[20];
8      char password[20];
9  }ADMINISTRATOR;
10
11 typedef struct user
12 {
13     int pakey;
14     char name[50];
15     char password[50];
16 }USER;
17
18 typedef struct passenger{
19     int pakey;
20     char ID[20];
21     char tel[20];
22     int sex;           // 1 男 2 女
23     int age;
24     int status;       //初始化为0，0为健康，1为有症状感染，2为无症状感染，3为密
接
25 }PASSENGER;
26
27 typedef struct record
28 {
29     int pakey;
30     int trkey;
31     char track[6];
32 }RECORD;
33
34 typedef struct train
35 {
36     int trkey;
37     char trainname[10];
38     int date;
```

```

38     int date;
39     int status;
40 }TRAIN;
41
42 typedef struct pagetrain{
43     int trkey;
44     char trainname[10];
45     int date;
46     int status;
47 }PAGETRAIN;
48
49 #endif

```

1. administrator

变量名	类型	解释说明
adkey	int	存储管理员的键
name	char[]	存储管理员用户名
password	char[]	存储管理员密码

2. user

变量名	类型	解释说明
pakey	int	存储乘客的键
name	char[]	存储乘客用户名
password	char[]	存储乘客密码

3. passenger

变量名	类型	解释说明
pakey	int	存储乘客的键
ID	char[]	存储乘客身份证号
tel	char[]	存储乘客电话号码

sex	int	存储乘客性别 1为男性，2为女性
age	int	存储乘客年龄
status	int	存储乘客健康状态 0为健康，1为有症状 2为无症状，3为密接

4. record

变量名	类型	解释说明
pakey	int	存储对应乘客的键
trkey	int	存储对应车次的键
track	char[]	存储详细行程轨迹

5. train

变量名	类型	解释说明
trkey	int	存储车次的键
trainname	char[]	存储车次名称
date	int	存储日期，date为距2020.1.1的天数
status	int	存储车次状态，0为正常，1为异常

6. pagetrain

变量名	类型	解释说明
trkey	int	存储车次的键
trainname	char[]	存储车次名称
date	int	存储日期，date为距2020.1.1的天

		数
status	int	存储车次状态，0为正常，1为异常

八、算法说明

- 1. 分页器功能
- 2. 基于动态规划获取最小相同字串长度的搜索功能

九、函数原型

1. config.h

C

```
1  #ifndef _config_h_
2  #define _config_h_
3
4  int judgeID(char *ID);
5  int judgetel(char *tel);
6  int leapyear(int yy);
7  int getdate(int date,int *yy,int *mm,int *dd);
8  int postdate(int yy,int mm,int dd,int *date);
9  char *datestring(int date);
10 int getsexbyID(char ID[20]);
11 int getagebyID(char ID[20]);
12 void gettodaydate(int *yy,int *mm,int *dd);
13 int xpos(char c);
14 int ypos(char c);
15 void gettrackstring(char *trainname,int i,char *desttrack);
16 int samestringmax(char *a,char *b);
17
18 #endif
```

2. control.h

C

```
1  #ifndef _control_h_
2  #define control h
```

```
3
4 int inbarword16(int left,int top,int barcolor,int width,int n,char *s,int word
  color);
5 int pressbarword16(int left,int top,int barcolor,int width,int n,char *s,int w
  ordcolor);
6
7 int inbarword24(int left,int top,int barcolor,int width,int n,char *s,int word
  color);
8 int pressbarword24(int left,int top,int barcolor,int width,int n,char *s,int w
  ordcolor);
9
10 int inbarword32(int left,int top,int barcolor,int width,int n,char *s,int word
  color);
11 int pressbarword32(int left,int top,int barcolor,int width,int n,char *s,int w
  ordcolor);
12
13 int inbarword48(int left,int top,int barcolor,int width,int n,char *s,int word
  color);
14 int pressbarword48(int left,int top,int barcolor,int width,int n,char *s,int w
  ordcolor);
15
16 int inbarwordframe(int left,int top,int barcolor,int width,char *s,int wordcol
  or,int framecolor);
17 int pressbarwordframe(int left,int top,int barcolor,int width,char *s,int word
  color,int framecolor);
18
19 int incircle(int x,int y,int radius);
20 int presscircle(int x,int y,int radius);
21
22 int inreturnbutton(void);
23 int pressreturnbutton(void);
24
25 int insearchbuttuo(void);
26 int presssearchbutton(void);
27
28 int inpalabelu(void);
29 int presspalabelu(void);
30
31 int inpalabelm(void);
32 int presspalabelm(void);
33
34 int inpalabell(void);
35 int presspalabell(void);
36
37 int inleftarrow(void);
38 int pressleftarrow(void);
39
```

```
40 int inrightarrow(void);
41 int pressrightarrow(void);
42
43 int intrack1(void);
44 int presstrack1(void);
45
46 int intrack2(void);
47 int presstrack2(void);
48
49 int intrack3(void);
50 int presstrack3(void);
51
52 int intrack4(void);
53 int presstrack4(void);
54
55 int intrack5(void);
56 int presstrack5(void);
57
58 int intrack6(void);
59 int presstrack6(void);
60
61 int intrack7(void);
62 int presstrack7(void);
63
64 int intrack8(void);
65 int presstrack8(void);
66
67 int intrack9(void);
68 int presstrack9(void);
69
70 int intrack10(void);
71 int presstrack10(void);
72
73 #endif
```

3. drawpage.h

C

```
1  #ifndef _DRAWPAGE_H_
2  #define _DRAWPAGE_H_
3
4  void drawstart();
5  void drawadlogin();
6  void drawpallogin();
7  void drawparegister();
8  void drawpabind();
9  void drawpamessage(struct passenger *pa);
10 void drawpapostrecord();
11 void drawpagetrecord(struct pagetrain *ptrnu,struct pagetrain *ptrnm,struct pagetrain *ptrnl,int currentpage,int countpage);
12 void drawpagettrack();
13 void drawstationselect(char *trainname);
14 void drawadlabel();
15 void drawadmanager();
16 void drawadhealth();
17 void drawadgetpospa(struct passenger *pau,struct passenger *pam,struct passenger *pal,int currentpage,int countpage);
18 void drawadgetcttpa(struct passenger *pau,struct passenger *pam,struct passenger *pal,int currentpage,int countpage);
19 void drawadgetpostr(struct pagetrain *ptrnu,struct pagetrain *ptrnm,struct pagetrain *ptrnl,int currentpage,int countpage);
20 void drawadsearch(struct passenger *pau,struct passenger *pam,struct passenger *pal,int currentpage,int countpage);
21
22 #endif
```

4. file.h

C

```
1  #ifndef _file_h_
2  #define _file_h_
3
4  void initfile(void);
5  int writeuser(char *name, char *password);
6  int loginuser(char *name, char *password);
7  int existusername(char *name);
8  int writeadmin(char *name, char *password);
9  int loginadmin(char *name, char *password);
10 int writepassenger(int pakey, char *ID, char *tel, int sex, int age, int status);
11 int existpassengerID(char *ID);
12 int getpassengerbystatus(int status, struct passenger destpa[50]);
13 int getpassengerbysearch(char *searchID, struct passenger destpa[20]);
14 int getpassengerbypakey(int pakey, struct passenger *destpa);
15 int getpakeybyID(char *ID, int *pakey);
16 int updatestatusbypakey(int pakey, int status);
17 int writetrain(char *trainname, int yy, int mm, int dd);
18 int gettrainbytrkey(int trkey, struct train *desttrn);
19 int gettrainbystatus(int status, struct pagetrain desttrn[20]);
20 int updatestatusbytrkey(int trkey, int status);
21 int getpostedtrkey(int date, char *trainname, int *trkey);
22
23 #endif
```

5. file2.h

C

```
1  #ifndef _file2_h_
2  #define _file2_h_
3
4  int writerecord(int pakey, int trkey);
5  int writerecordv1(int pakey, int trkey, char *track);
6  int getrecordbypakey(int pakey, int desttrkeyset[50]);
7  int getrecordbytrkey(int trkey, int destpakeyset[50]);
8  int gettrackbypakey(int pakey, char trackset[10][10]);
9
10 #endif
```

6. filetxt.h

C++

```
1  #ifndef _filetxt_h_
2  #define _filetxt_h_
3
4  int getcnamebyID(char *ID, char *cname);
5  int getstartbytrainname(char *trainname, char *start);
6  int getendbytrainname(char *trainname, char *end);
7
8  #endif
```

7. graphpro.h

C

```
1  #ifndef _GRAPHPRO_H_
2  #define _GRAPHPRO_H_
3
4  void drawtop(void);
5  void barword16(int left, int top, int barcolor, int width, int n, char *s, int wordcolor);
6  void barword24(int left, int top, int barcolor, int width, int n, char *s, int wordcolor);
7  void barword32(int left, int top, int barcolor, int width, int n, char *s, int wordcolor);
8  void barword48(int left, int top, int barcolor, int width, int n, char *s, int wordcolor);
9  void barwordframe(int left, int top, int barcolor, int n, char *s, int wordcolor, int framecolor);
10 void returnbutton(int color);
11 void searchbutton(int color);
12 void draw_date(int x, int y, int wordcolor);
13 void loginok(void);
14 void loginfail(void);
15 void loginpasswordwrong(void);
16 void loginnoregister(void);
17 void registerok(void);
18 void registerrepeat(void);
19 void registerpasswrong(void);
20 void registerpasszero(void);
21 void registernamezero(void);
22 void registerpassshort(void);
23 void bindok(void);
24 void bindIDrepeat(void);
25 void bindIDwrong(void);
26 void bindtelwrong(void);
```

```
27 void labelok(void);
28 void labelIDzero(void);
29 void statusabnormal(void);
30 void recordzero(void);
31 void trackrecordrepeat(void);
32 void trackrecordok(void);
33 void palabelu(struct passenger *pa);
34 void palabelm(struct passenger *pa);
35 void palabell(struct passenger *pa);
36 void trlabelu(struct pagetrain *trn);
37 void trlabelm(struct pagetrain *trn);
38 void trlabell(struct pagetrain *trn);
39 void leftarrow(int color);
40 void rightarrow(int color);
41 void pagenumber(int currentpage, int countpage);
42 void putnum(int x, int y, int num);
43 void todaydate(void);
44 void recordmessage(char* trainname);
45 void position1(int x, int y, int *s);
46 void position2(int x, int y, int *s);
47 void drawmap(void);
48 void mapline(char *s, int color);
49 void track1(char *trainname, int color);
50 void track2(char *trainname, int color);
51 void track3(char *trainname, int color);
52 void track4(char *trainname, int color);
53 void track5(char *trainname, int color);
54 void track6(char *trainname, int color);
55 void track7(char *trainname, int color);
56 void track8(char *trainname, int color);
57 void track9(char *trainname, int color);
58 void track10(char *trainname, int color);
59
60 #endif
```

8. page2.h

C++

```
1  #ifndef _page2_h_
2  #define _page2_h_
3
4  void pagestart(int *page);
5  void pagepalogin(int *page, int *pakey);
6  void pageadlogin(int *page);
7
8  #endif
```

9. pagead.h

C++

```
1  #ifndef _pagead_h_
2  #define _pagead_h_
3
4  void pageadmanager(int *page, char searchID[20]);
5  void pageadlabel(int *page);
6  void pageadgetpospa(int *page);
7  void pageadgetcttpa(int *page);
8  void pageadgetpostrn(int *page);
9  void pageadsearch(int *page, char searchID[20], int *pakey);
10 void pageadpamessage(int *page, int *pakey);
11 void pageadpagetrack(int *page, int *pakey);
12 void pageadpagetrecord(int *page, int *pakey);
13
14 #endif
```

10. pagepa.h

C

```
1  #ifndef _pagepa_h_
2  #define _pagepa_h_
3
4  void pageparegister(int *page);
5  void pagepabind(int *page, int *pakey);
6  void pagepamessage(int *page, int *pakey);
7  void pagepapostrecord(int *page, int *pakey, int *trkey);
8  void pagepagetrecord(int *page, int *pakey);
9  void pageposttrack(int *page, int *pakey, int *trkey);
10 void pagepagetrack(int *page, int *pakey);
11
12 #endif
```

11. public.h

C

```
1  #ifndef _public_h_
2  #define _public_h_
3
4  #include<graphics.h>
5  #include<time.h>
6  #include<bios.h>
7  #include<stdio.h>
8  #include<stdlib.h>
9  #include<string.h>
10 #include "hz.h"
11 #include "graphpro.h"
12 #include "mouse.h"
13 #include "control.h"
14 #include "drawpage.h"
15 #include "input.h"
16 #include "page2.h"
17 #include "pagead.h"
18 #include "pagepa.h"
19 #include "data.h"
20 #include "file.h"
21 #include "file2.h"
22 #include "filetxt.h"
23 #include "config.h"
24
25 #endif
```

12. hz.h

C++

```
1  #ifndef __HZ_H__
2  #define __HZ_H__
3
4  void puthz(int x, int y, char *s, int flag, int part, int color);
5
6  #endif
```

13. input.h

C

```
1  #ifndef _input_h_
2  #define _input_h_
3
4  void Input_Vis(char* ip, int x, int y, int lim, int color, int wordcolor);
5  void Input_Invis(char* ip, int x, int y, int lim, int color, int wordcolor);
6
7  #endif
```

14. mouse.h

C

```
1  #ifndef _mouse_h_
2  #define _mouse_h_
3
4  int mouse_press(int x1, int y1, int x2, int y2); //如果在框中点击, 则返回1; 在框中
    未点击, 则返回2; 不在框中则返回0
5  void mouse(int,int); //设计鼠标
6  void mouseinit(void); //初始化
7  //void mou_pos(int*,int*,int*); //更改鼠标位置
8  void mread(int *,int *,int*); //改坐标不画
9  void save_bk_mou(int x,int y); //存鼠标背景
10 void clrmous(int x,int y); //清除鼠标
11 void drawmous(int x,int y); //画鼠标
12 void newmouse(int *nx,int *ny,int *nbuttons); //更新鼠标
13
14 extern int MouseX;
15 extern int MouseY;
16 extern int MouseS;
17 extern int press;
18 extern union REGS regs;
19 #endif
```

十、感想体会

1. 李洋

日月如梭，白驹过隙，c语言课程设计的日子很快到来了。回想过去的几个月不禁让我们感慨万分，初开始面对c课设的迷惘无助仿佛还在昨天。从大一初入学校就开始听说关于c课设的种种传闻与故事，曾经只是轻松一笑，到我们亲自来完成这份任务时却才感受到了真正的压力与紧张感。初学c课设的我完全无法将课本与程序设计相结合，无从下手的我焦急万分、不知所措，好在许多学长学姐和我的队友为我提供了很多的帮助，我在慢慢摸索中一步步学习c语言程序设计。c课设是一项好大的工程，初次面对它让我有些喘不过气，但在一步步前进中，我跌跌撞撞的迈上了c语言程序设计这条艰巨的路。

在设计之初，我纠结与课本内容的讲解与学习，但在一步步摸索中，我试着在书写代码的过程中更好地领悟c课设的构架与逻辑。我和队友开始对c课设进行第一步的需求分析，在需求分析后，我们逐步确立了自己程序的设计方向与大体思路，我们又从网络上参考铁路12306等相关的app与网站，从更多方面考虑用户会想要使用一个包含什么样功能的系统来便利自己的生活。接着我参考了一些学长学姐的代码，对程序设计的框架有了一个更加明确直观的认识，从图形界面的绘制到界面跳转的完成，从按钮功能的实现到文件结构体的完善。在代码结构逐步完善的同时，我们的信心也不断增强。

同时我的队友帮助我开始使用git进行线上的代码传输，使得我们能随时随地共享我们更新的代码。c课设是一项浩大的工程，在c课设的完成过程中我更深的领悟到了团队合作的魅力与重要性，在队友的帮助下我们共同努力共同进步，虽然时刻有压力伴在心中，但在鼓励与坚持中我们共同进步，不断地攻克难关，我在不断地激励自己不能拖对方后腿，同时有问题我们也会共同努力共同将其解决。

课设让我对问题的解决有了更深一步的了解。之前在学习c语言的过程中，代码出现问题后便通过系统提供的报错来对代码进行修改与完善，但在c语言课程设计的过程中，许多问题不会出现在error之中，这就需要通过调试来对bug进行修复与完善，刚开始的我对此一筹莫展，只能通过目力来寻找可能存在的错误与问题，但在逐渐地对调试的学习中，我渐渐学会了从问题出现打大概位置对代码进行调试以进行进一步的修改，尽管有时候会被一个不知名的bug卡住进度许久，但在bug解决运行成功的那一瞬间所带来的喜悦比设计程序带来的更为强烈。

c课设在为我们带来压力的同时也带来了动力，这是一项很考验我意志力与坚持力的艰巨任务，c课设在占用了大量时间的同时极为考验我们对学习与编写程序时间的平衡能力，尤其是在面临课设验收和微积分，物理期中考试同时到来的节点上，更为考验我们对时间的管理能力和我们对压力的处理能力。在巨大的压力之下，我内心也曾有过灰心与恐惧，但幸运的是，我的队友为我提供了巨大的帮助，在他的帮助下，我不断处理各方面压力，不断将其均衡。

这次c课设让我学到了很多，无论是需求分析，对bug的处理能力，还是团队合作完成任务的过程都让我受益匪浅。

2. 张子陆

马上就要迎来c课设的验收，而为一个bug苦苦皱眉一晚的日子仿佛还在昨天，c课设完成的过程中，有收获，有压力，有喜悦，也有痛苦，但我们还是坚持走到了最后一步。一路坚持走来的收获最多的也许不是代码的学习与理解，更大的收获是在克服困难路上的坚持与不断挑战自我的突破，c课设的完成让我收益颇多，让我在突破中成长了自我。

开始做c课设的时候遇到了许多困难，无法合理分配课业学习与c课设的时间安排，

十一、源代码

1. main.c

C

```
1  #include"public.h"
2
3  int main(){
4      int gd =VGA;
5      int gm =VGAHI;
6      char S[30];
```

```
7     int page=0;
8     FILE *log;
9     int pakeymain;
10    int trkeymain;
11    int i;
12    char searchID[20];
13    struct train trainmain1;
14    struct train trainmain2;
15    int yy,mm,dd;
16    int trkeyset[50];
17    int pakeyset[50];
18    char trackset[10][10];
19
20    initfile();
21    initgraph(&gd,&gm,".\\BGI");
22    mouseinit();
23    setcolor(BLUE);
24
25    while (1)
26    {
27        switch (page)
28        {
29            case 0:
30                pagestart(&page);
31                break;
32            case 1:
33                pagepalogin(&page,&pakeymain);
34                break;
35            case 2:
36                pageadlogin(&page);
37                break;
38            case 3:
39                pageparegister(&page);
40                break;
41            case 4:
42                pagepabind(&page,&pakeymain);
43                break;
44            case 5:
45                pagepamessage(&page,&pakeymain);
46                break;
47            case 6:
48                pagepapostrecord(&page,&pakeymain,&trkeymain);
49                break;
50            case 7:
51                pagepagetrecord(&page,&pakeymain);
52                break;
53            case 8:
54                pageposttrack(&page,&pakeymain,&trkeymain);
```

```

55         break;
56     case 9:
57         pagepagetrack(&page,&pakeymain);
58         break;
59     case 21:
60         pageadmanager(&page,searchID);
61         break;
62     case 211:
63         pageadlabel(&page);
64         break;
65     case 212:
66         pageadgetpospa(&page);
67         break;
68     case 213:
69         pageadgetpostrn(&page);
70         break;
71     case 214:
72         pageadgetccttpa(&page);
73         break;
74     case 215:
75         pageadsearch(&page,searchID);
76         break;
77     }
78 }
79 }

```

2. config.c

C

```

1  int judgetel(char *tel){
2      if(strlen(tel)!=11){
3          return 0;
4      }
5      if(*tel!='1'){
6          return 0;
7      }
8      return 1;
9  }
10
11 int leapyear(int yy){
12     if((yy%4==0&&yy%100!=0) || (yy%400==0))return 1;
13     else return 0;
14 }
15
16 int getdate(int date,int *yy,int *mm,int *dd){

```

```
17     int year=2020;
18     int month;
19     int day;
20
21     if(date<=0){
22         return 0;
23     }
24
25     date++;
26     while(date>365){
27         if(leapyear(year)==1&&date>366){
28             date-=366;
29             year++;
30         }
31         else if(leapyear(year)!=1){
32             date-=365;
33             year++;
34         }
35     }
36     if(leapyear(year)==1){
37         if(date>=1&&date<=31){
38             month=1;
39             day=date-0;
40         }
41         if(date>=32&&date<=60){
42             month=2;
43             day=date-31;
44         }
45         if(date>=61&&date<=91){
46             month=3;
47             day=date-60;
48         }
49         if(date>=92&&date<=121){
50             month=4;
51             day=date-91;
52         }
53         if(date>=122&&date<=152){
54             month=5;
55             day=date-121;
56         }
57         if(date>=153&&date<=182){
58             month=6;
59             day=date-152;
60         }
61         if(date>=183&&date<=213){
62             month=7;
63             day=date-182;
64         }
```

```
65         if(date>=214&&date<=244){
66             month=8;
67             day=date-213;
68         }
69         if(date>=245&&date<=274){
70             month=9;
71             day=date-244;
72         }
73         if(date>=275&&date<=305){
74             month=10;
75             day=date-274;
76         }
77         if(date>=306&&date<=335){
78             month=11;
79             day=date-305;
80         }
81         if(date>=336&&date<=366){
82             month=12;
83             day=date-335;
84         }
85     }
86     else{
87         if(date>=1&&date<=31){
88             month=1;
89             day=date-0;
90         }
91         if(date>=32&&date<=59){
92             month=2;
93             day=date-31;
94         }
95         if(date>=60&&date<=90){
96             month=3;
97             day=date-59;
98         }
99         if(date>=91&&date<=120){
100             month=4;
101             day=date-90;
102         }
103         if(date>=121&&date<=151){
104             month=5;
105             day=date-120;
106         }
107         if(date>=152&&date<=181){
108             month=6;
109             day=date-151;
110         }
111         if(date>=182&&date<=212){
112             month=7;
```

```

112         month--;
113         day=date-181;
114     }
115     if(date>=213&&date<=243){
116         month=8;
117         day=date-212;
118     }
119     if(date>=244&&date<=273){
120         month=9;
121         day=date-243;
122     }
123     if(date>=274&&date<=304){
124         month=10;
125         day=date-273;
126     }
127     if(date>=305&&date<=334){
128         month=11;
129         day=date-304;
130     }
131     if(date>=335&&date<=365){
132         month=12;
133         day=date-334;
134     }
135 }
136 *yy=year;
137 *mm=month;
138 *dd=day;
139 return 1;
140 }
141
142 int postdate(int yy,int mm,int dd,int *date){
143     if(yy>=2028||yy<=2019){
144         *date=-1;
145         return 0;
146     }
147     if(mm<=0||mm>=13){
148         *date=-1;
149         return 0;
150     }
151     if(leapyear(yy)==1&&mm==2){
152         if(dd<=0||dd>=30){
153             *date=-1;
154             return 0;
155         }
156     }
157     if(leapyear(yy)!=1&&mm==2){
158         if(dd<=0||dd>=29){
159             *date=-1;

```

```
160         return 0;
161     }
162 }
163 if(mm==1 | mm==3 | mm==5 | mm==7 | mm==8 | mm==10 | mm==12){
164     if(dd<=0 | dd>=32){
165         *date=-1;
166         return 0;
167     }
168 }
169 if(mm==4 | mm==6 | mm==9 | mm==11){
170     if(dd<=0 | dd>=31){
171         *date=-1;
172         return 0;
173     }
174 }
175
176 *date=0;
177 *date+=dd;
178 mm--;
179 while(mm>0){
180     if(mm==1 | mm==3 | mm==5 | mm==7 | mm==8 | mm==10 | mm==12){
181         *date+=31;
182         mm--;
183     }
184     if(mm==4 | mm==6 | mm==9 | mm==11){
185         *date+=30;
186         mm--;
187     }
188     if(leapyear(yy)==1&&mm==2){
189         *date+=29;
190         mm--;
191     }
192     if(leapyear(yy)!=1&&mm==2){
193         *date+=28;
194         mm--;
195     }
196 }
197 while(yy>2020){
198     if(leapyear(yy)==1){
199         *date+=366;
200         yy--;
201     }
202     else{
203         *date+=365;
204         yy--;
205     }
206 }
207 return 1;
```

```

208 }
209
210 char *datestring(int date){
211     int yy,mm,dd;
212     char ystring[5];
213     char mstring[5];
214     char dstring[5];
215     char *dests;
216
217     memset(ystring,'\0',sizeof(ystring));
218     memset(mstring,'\0',sizeof(mstring));
219     memset(dstring,'\0',sizeof(dstring));
220
221     getdate(date,&yy,&mm,&dd);
222
223     itoa(yy,ysstring,10);
224     itoa(mm,mstring,10);
225     itoa(dd,dstring,10);
226
227     //todo:!!!
228     strcpy(dests,ysstring);
229     strcat(dests,".");
230     strcat(dests,mstring);
231     strcat(dests,".");
232     strcat(dests,dstring);
233
234     return dests;
235 }
236
237 int getsexbyID(char ID[20]){
238     int x;
239
240     x=ID[16]-'0';
241     if(x%2==0){
242         return 2;
243     }
244     else{
245         return 1;
246     }
247 }
248
249 int getagebyID(char ID[20]){
250     int age;
251     int IDyy,IDmm,IDdd;
252     int yy,mm,dd;
253     gettodaydate(&yy,&mm,&dd);
254     IDyy=(ID[6]-'0')*1000+(ID[7]-'0')*100+(ID[8]-'0')*10+(ID[9]-'0');
255     IDmm=(ID[10]-'0')*10+(ID[11]-'0');

```



```

255     IDmm=(ID[10]-'0')*10+(ID[11]-'0');
256     IDdd=(ID[12]-'0')*10+(ID[13]-'0');
257
258     age=yy-IDyy;
259     if(mm<IDmm)age--;
260     if(mm==IDmm&&dd<IDdd)age--;
261
262     return age;
263 }
264
265 void gettodaydate(int *yy,int *mm,int *dd){
266     struct tm* ptr;
267     time_t lt;
268
269     time(&lt);
270     ptr = localtime(&lt);
271
272
273     *yy=ptr->tm_year + 1900;
274     *mm=ptr->tm_mon + 1;
275     *dd=ptr->tm_mday;
276
277     *yy=2022;
278     *mm=4;
279     *dd=17;
280
281     return;
282 }
283
284 int xpos(char c){
285     if(c=='a')return 425;
286     if(c=='b')return 325;
287     if(c=='c')return 155;
288     if(c=='d')return 55;
289     if(c=='e')return 95;
290     if(c=='f')return 305;
291     if(c=='g')return 465;
292     if(c=='h')return 485;
293     if(c=='i')return 505;
294     if(c=='j')return 565;
295     if(c=='k')return 209;
296     if(c=='l')return 229;
297     if(c=='m')return 385;
298     if(c=='n')return 200;
299 }
300
301 int ypos(char c){
302     if(c=='a')return 120;

```

```

303     if(c=='b')return 165;
304     if(c=='c')return 210;
305     if(c=='d')return 270;
306     if(c=='e')return 310;
307     if(c=='f')return 270;
308     if(c=='g')return 240;
309     if(c=='h')return 280;
310     if(c=='i')return 250;
311     if(c=='j')return 252;
312     if(c=='k')return 420;
313     if(c=='l')return 440;
314     if(c=='m')return 330;
315     if(c=='n')return 350;
316 }
317
318 void gettrackstring(char *trainname,int i,char *desttrack){
319     if(strcmp(trainname,"G562")==0){
320         if(i==1)strcpy(desttrack,"ab");return;
321         if(i==2)strcpy(desttrack,"abg");return;
322         if(i==3)strcpy(desttrack,"abgi");return;
323         if(i==4)strcpy(desttrack,"abgij");return;
324         if(i==5)strcpy(desttrack,"bg");return;
325         if(i==6)strcpy(desttrack,"bgi");return;
326         if(i==7)strcpy(desttrack,"bgij");return;
327         if(i==8)strcpy(desttrack,"gi");return;
328         if(i==9)strcpy(desttrack,"gij");return;
329         if(i==10)strcpy(desttrack,"ij");return;
330     }
331     if(strcmp(trainname,"G567")==0){
332         if(i==1)strcpy(desttrack,"jh");return;
333         if(i==2)strcpy(desttrack,"jhm");return;
334         if(i==3)strcpy(desttrack,"jhml");return;
335         if(i==4)strcpy(desttrack,"jhmlk");return;
336         if(i==5)strcpy(desttrack,"hm");return;
337         if(i==6)strcpy(desttrack,"hml");return;
338         if(i==7)strcpy(desttrack,"hmlk");return;
339         if(i==8)strcpy(desttrack,"ml");return;
340         if(i==9)strcpy(desttrack,"mlk");return;
341         if(i==10)strcpy(desttrack,"lk");return;
342     };
343     if(strcmp(trainname,"G751")==0){
344         if(i==1)strcpy(desttrack,"mf");return;
345         if(i==2)strcpy(desttrack,"mfg");return;
346         if(i==3)strcpy(desttrack,"mfgb");return;
347         if(i==4)strcpy(desttrack,"mfgba");return;
348         if(i==5)strcpy(desttrack,"fg");return;
349         if(i==6)strcpy(desttrack,"fgb");return;
350         if(i==7)strcpy(desttrack,"fgba");return;

```

```

351     if(i==8)strcpy(desttrack,"gb");return;
352     if(i==9)strcpy(desttrack,"gba");return;
353     if(i==10)strcpy(desttrack,"ba");return;
354 }
355 if(strcmp(trainname,"G768")==0){
356     if(i==1)strcpy(desttrack,"ke");return;
357     if(i==2)strcpy(desttrack,"ked");return;
358     if(i==3)strcpy(desttrack,"kedc");return;
359     if(i==4)strcpy(desttrack,"kedcb");return;
360     if(i==5)strcpy(desttrack,"ed");return;
361     if(i==6)strcpy(desttrack,"edc");return;
362     if(i==7)strcpy(desttrack,"edcb");return;
363     if(i==8)strcpy(desttrack,"dc");return;
364     if(i==9)strcpy(desttrack,"dcb");return;
365     if(i==10)strcpy(desttrack,"cb");return;
366 }
367 if(strcmp(trainname,"G267")==0){
368     if(i==1)strcpy(desttrack,"lk");return;
369     if(i==2)strcpy(desttrack,"lkn");return;
370     if(i==3)strcpy(desttrack,"lknf");return;
371     if(i==4)strcpy(desttrack,"lknfa");return;
372     if(i==5)strcpy(desttrack,"kn");return;
373     if(i==6)strcpy(desttrack,"knf");return;
374     if(i==7)strcpy(desttrack,"knfa");return;
375     if(i==8)strcpy(desttrack,"nf");return;
376     if(i==9)strcpy(desttrack,"nfa");return;
377     if(i==10)strcpy(desttrack,"fa");return;
378 }
379 if(strcmp(trainname,"G186")==0){
380     if(i==1)strcpy(desttrack,"gf");return;
381     if(i==2)strcpy(desttrack,"gfn");return;
382     if(i==3)strcpy(desttrack,"gfne");return;
383     if(i==4)strcpy(desttrack,"gfned");return;
384     if(i==5)strcpy(desttrack,"fn");return;
385     if(i==6)strcpy(desttrack,"fne");return;
386     if(i==7)strcpy(desttrack,"fned");return;
387     if(i==8)strcpy(desttrack,"ne");return;
388     if(i==9)strcpy(desttrack,"ned");return;
389     if(i==10)strcpy(desttrack,"ed");return;
390 }
391 if(strcmp(trainname,"G379")==0){
392     if(i==1)strcpy(desttrack,"hm");return;
393     if(i==2)strcpy(desttrack,"hmf");return;
394     if(i==3)strcpy(desttrack,"hmfcd");return;
395     if(i==4)strcpy(desttrack,"hmfc");return;
396     if(i==5)strcpy(desttrack,"mf");return;
397     if(i==6)strcpy(desttrack,"mfc");return;

```

```

398     if(i==7)strcpy(desttrack,"mfc");return;
399     if(i==8)strcpy(desttrack,"fc");return;
400     if(i==9)strcpy(desttrack,"fcd");return;
401     if(i==10)strcpy(desttrack,"cd");return;
402 }
403 if(strcmp(trainname,"G467")==0){
404     if(i==1)strcpy(desttrack,"cf");return;
405     if(i==2)strcpy(desttrack,"cfm");return;
406     if(i==3)strcpy(desttrack,"cfmh");return;
407     if(i==4)strcpy(desttrack,"cfmhj");return;
408     if(i==5)strcpy(desttrack,"fm");return;
409     if(i==6)strcpy(desttrack,"fmh");return;
410     if(i==7)strcpy(desttrack,"fmhj");return;
411     if(i==8)strcpy(desttrack,"mh");return;
412     if(i==9)strcpy(desttrack,"mhj");return;
413     if(i==10)strcpy(desttrack,"hj");return;
414 }
415 if(strcmp(trainname,"G685")==0){
416     if(i==1)strcpy(desttrack,"im");return;
417     if(i==2)strcpy(desttrack,"ima");return;
418     if(i==3)strcpy(desttrack,"imab");return;
419     if(i==4)strcpy(desttrack,"imabc");return;
420     if(i==5)strcpy(desttrack,"ma");return;
421     if(i==6)strcpy(desttrack,"mab");return;
422     if(i==7)strcpy(desttrack,"mabc");return;
423     if(i==8)strcpy(desttrack,"ab");return;
424     if(i==9)strcpy(desttrack,"abc");return;
425     if(i==10)strcpy(desttrack,"bc");return;
426 }
427 if(strcmp(trainname,"G335")==0){
428     if(i==1)strcpy(desttrack,"af");return;
429     if(i==2)strcpy(desttrack,"afn");return;
430     if(i==3)strcpy(desttrack,"afnk");return;
431     if(i==4)strcpy(desttrack,"afnkl");return;
432     if(i==5)strcpy(desttrack,"fn");return;
433     if(i==6)strcpy(desttrack,"fnk");return;
434     if(i==7)strcpy(desttrack,"fnkl");return;
435     if(i==8)strcpy(desttrack,"nk");return;
436     if(i==9)strcpy(desttrack,"nkl");return;
437     if(i==10)strcpy(desttrack,"kl");return;
438 }
439 }
440
441 int samestringmax(char *a,char *b){
442     int lengtha=0;
443     int lengthb=0;
444     char atemp[20]={'\0'};
445     char btemp[20]={'\0'};

```

```

446     int res[20][20]={0};
447     int maxres=0;
448     int i,j;
449
450     while(*a!='\0'){
451         atemp[lengtha]=*a;
452         a++;
453         lengtha++;
454     }
455     while(*b!='\0'){
456         btemp[lengthb]=*b;
457         b++;
458         lengthb++;
459     }
460
461     for(i=0;i<lengtha;i++){
462         for(j=0;j<lengthb;j++){
463             if(atemp[i]==btemp[j]){
464                 res[i+1][j+1]=res[i][j]+1;
465                 maxres=max(maxres,res[i+1][j+1]);
466             }
467         }
468     }
469
470     return maxres;
471 }

```

3. control.c

C++

```

1  #include"mouse.h"
2
3  int inbarword16(int left,int top,int barcolor,int width,int n,char *s,int word
color){
4      int right=width*16+4+left;
5      int bottom=20+top;
6
7      if(left==0){
8          if(MouseX>320-8*width-2&&MouseY>top&&MouseX<320+8*width+2&&MouseY<bott
om)return 1;
9          else return 0;
10     }
11
12     if(MouseX>left&&MouseY>top&&MouseX<right&&MouseY<bottom)return 1;
13     else return 0;

```

```

14 }
15 int pressbarword16(int left,int top,int barcolor,int width,int n,char *s,int w
ordcolor){
16     int right=width*16+4+left;
17     int bottom=20+top;
18
19     if(left==0){
20         if(mouse_press(320-8*width-2,top,320+8*width+2,bottom)==1)return 1;
21         else if(mouse_press(320-8*width-2,top,320+8*width+2,bottom)==2)return
2;
22         else return 0;
23     }
24     if(mouse_press(left,top,right,bottom)==1)return 1;
25     else if(mouse_press(left,top,right,bottom)==2)return 2;
26     else return 0;
27 }
28
29 int inbarword24(int left,int top,int barcolor,int width,int n,char *s,int word
color){
30     int right=width*24+6+left;
31     int bottom=30+top;
32
33     if(left==0){
34         if(MouseX>320-12*width-3&&MouseY>top&&MouseX<320+12*width+3&&MouseY<bo
ttom)return 1;
35         else return 0;
36     }
37
38     if(MouseX>left&&MouseY>top&&MouseX<right&&MouseY<bottom)return 1;
39     else return 0;
40 }
41 int pressbarword24(int left,int top,int barcolor,int width,int n,char *s,int w
ordcolor){
42     int right=width*24+6+left;
43     int bottom=30+top;
44
45     if(left==0){
46         if(mouse_press(320-12*width-3,top,320+12*width+3,bottom)==1)return 1;
47         else if(mouse_press(320-12*width-3,top,320+12*width+3,bottom)==2)retur
n 2;
48         else return 0;
49     }
50     if(mouse_press(left,top,right,bottom)==1)return 1;
51     else if(mouse_press(left,top,right,bottom)==2)return 2;
52     else return 0;
53 }
54
55 int inbarword32(int left,int top,int barcolor,int width,int n,char *s,int word

```

```

color){
56     int right=width*32+8+left;
57     int bottom=40+top;
58
59     if(left==0){
60         if(MouseX>320-16*width-4&&MouseY>top&&MouseX<320+16*width+4&&MouseY<bo
ttom)return 1;
61         else return 0;
62     }
63
64     if(MouseX>left&&MouseY>top&&MouseX<right&&MouseY<bottom)return 1;
65     else return 0;
66 }
67 int pressbarword32(int left,int top,int barcolor,int width,int n,char *s,int w
ordcolor){
68     int right=width*32+8+left;
69     int bottom=40+top;
70
71     if(left==0){
72         if(mouse_press(320-16*width-4,top,320+16*width+4,bottom)==1)return 1;
73         else if(mouse_press(320-16*width-4,top,320+16*width+4,bottom)==2)retur
n 2;
74         else return 0;
75     }
76     if(mouse_press(left,top,right,bottom)==1)return 1;
77     else if(mouse_press(left,top,right,bottom)==2)return 2;
78     else return 0;
79 }
80
81 int inbarword48(int left,int top,int barcolor,int width,int n,char *s,int word
color){
82     int right=width*48+12+left;
83     int bottom=60+top;
84
85     if(left==0){
86         if(MouseX>320-24*width-6&&MouseY>top&&MouseX<320+24*width+6&&MouseY<bo
ttom)return 1;
87         else return 0;
88     }
89
90     if(MouseX>left&&MouseY>top&&MouseX<right&&MouseY<bottom)return 1;
91     else return 0;
92 }
93 int pressbarword48(int left,int top,int barcolor,int width,int n,char *s,int w
ordcolor){
94     int right=width*48+12+left;
95     int bottom=60+top;
96

```

```

97     if(left==0){
98         if(mouse_press(320-24*width-6,top,320+24*width+6,bottom)==1)return 1;
99         else if(mouse_press(320-24*width-6,top,320+24*width+6,bottom)==2)retur
n 2;
100         else return 0;
101     }
102     if(mouse_press(left,top,right,bottom)==1)return 1;
103     else if(mouse_press(left,top,right,bottom)==2)return 2;
104     else return 0;
105 }
106
107 int inbarwordframe(int left,int top,int barcolor,int width,char *s,int wordcol
or,int framecolor){
108     int right=width*32+8+left;
109     int bottom=40+top;
110
111     if(left==0){
112         if(MouseX>320-16*width-4&&MouseY>top&&MouseX<320+16*width+4&&MouseY<bo
ttom)return 1;
113         else return 0;
114     }
115
116     if(MouseX>left&&MouseY>top&&MouseX<right&&MouseY<bottom)return 1;
117     else return 0;
118 }
119 int pressbarwordframe(int left,int top,int barcolor,int width,char *s,int word
color,int framecolorr){
120     int right=width*32+8+left;
121     int bottom=40+top;
122
123     if(left==0){
124         if(mouse_press(320-16*width-4,top,320+16*width+4,bottom)==1)return 1;
125         else if(mouse_press(320-16*width-4,top,320+16*width+4,bottom)==2)retur
n 2;
126         else return 0;
127     }
128     if(mouse_press(left,top,right,bottom)==1)return 1;
129     else if(mouse_press(left,top,right,bottom)==2)return 2;
130     else return 0;
131 }
132
133 int incircle(int x,int y,int radius){
134     int left=x-radius;
135     int right=x+radius;
136     int top=y-radius;
137     int bottom=y+radius;
138

```



```
139     if(MouseX>left&&MouseY>top&&MouseX<right&&MouseY<bottom)return 1;
140     else return 0;
141 }
142 int presscircle(int x,int y,int radius){
143     int left=x-radius;
144     int right=x+radius;
145     int top=y-radius;
146     int bottom=y+radius;
147
148     if(mouse_press(left,top,right,bottom)==1)return 1;
149     else if(mouse_press(left,top,right,bottom)==2)return 2;
150     else return 0;
151 }
152
153 int inreturnbutton(void){
154     if(MouseX>10&&MouseY>5&&MouseX<33&&MouseY<30)return 1;
155     else return 0;
156 }
157 int pressreturnbutton(void){
158     if(mouse_press(10,5,33,30)==1)return 1;
159     else if(mouse_press(10,5,33,30)==2)return 2;
160     else return 0;
161 }
162
163 int insearchbutton(void){
164     if(MouseX>520&&MouseY>100&&MouseX<550&&MouseY<140)return 1;
165     else return 0;
166 }
167 int presssearchbutton(void){
168     if(mouse_press(510,100,550,140)==1)return 1;
169     else if(mouse_press(510,100,550,140)==2)return 2;
170     else return 0;
171 }
172
173 int inpalabelu(void){
174     if(MouseX>60&&MouseY>130&&MouseX<580&&MouseY<210)return 1;
175     else return 0;
176 }
177 int presspalabelu(void){
178     if(mouse_press(60,130,580,210)==1)return 1;
179     else if(mouse_press(60,130,580,210)==2)return 2;
180     else return 0;
181 }
182
183 int inpalabelm(void){
184     if(MouseX>60&&MouseY>230&&MouseX<580&&MouseY<310)return 1;
185     else return 0;
186 }
```

```

187 int presspalabelm(void){
188     if(mouse_press(60,230,580,310)==1)return 1;
189     else if(mouse_press(60,230,580,310)==2)return 2;
190     else return 0;
191 }
192
193 int inpalabell(void){
194     if(MouseX>60&&MouseY>330&&MouseX<580&&MouseY<410)return 1;
195     else return 0;
196 }
197 int presspalabell(void){
198     if(mouse_press(60,330,580,410)==1)return 1;
199     else if(mouse_press(60,330,580,410)==2)return 2;
200     else return 0;
201 }
202
203 int inleftarrow(void){
204     if(MouseX>250&&MouseY>430&&MouseX<270&&MouseY<450)return 1;
205     else return 0;
206 }
207 int pressleftarrow(void){
208     if(mouse_press(250,430,270,450)==1)return 1;
209     else if(mouse_press(250,430,270,450)==2)return 2;
210     else return 0;
211 }
212
213 int inrightarrow(void){
214     if(MouseX>370&&MouseY>430&&MouseX<390&&MouseY<450)return 1;
215     else return 0;
216 }
217 int pressrightarrow(void){
218     if(mouse_press(370,430,390,450)==1)return 1;
219     else if(mouse_press(370,430,390,450)==2)return 2;
220     else return 0;
221 }
222
223 int intrack1(void){
224     if(MouseX>(110-3)&&MouseY>190&&MouseX<(110+24*6+3)&&MouseY<(190+24))return
225     1;
226     else return 0;
227 }
228 int presstrack1(void){
229     if(mouse_press(110,190,110+24*6+6,190+24)==1)return 1;
230     else if(mouse_press(110,190,110+24*6+6,190+24)==2)return 2;
231     else return 0;
232 }
233 int intrack2(void){

```

```

233 int intrack2(void){
234     if(MouseX>(110-3)&&MouseY>246&&MouseX<(110+24*6+3)&&MouseY<(246+24))return
1;
235     else return 0;
236 }
237 int presstrack2(void){
238     if(mouse_press(110,246,110+24*6+6,246+24)==1)return 1;
239     else if(mouse_press(110,246,110+24*6+6,246+24)==2)return 2;
240     else return 0;
241 }
242
243 int intrack3(void){
244     if(MouseX>(110-3)&&MouseY>302&&MouseX<(110+24*6+3)&&MouseY<(302+24))return
1;
245     else return 0;
246 }
247 int presstrack3(void){
248     if(mouse_press(110,302,110+24*6+6,302+24)==1)return 1;
249     else if(mouse_press(110,302,110+24*6+6,302+24)==2)return 2;
250     else return 0;
251 }
252
253 int intrack4(void){
254     if(MouseX>(110-3)&&MouseY>358&&MouseX<(110+24*6+3)&&MouseY<(358+24))return
1;
255     else return 0;
256 }
257 int presstrack4(void){
258     if(mouse_press(110,358,110+24*6+6,358+24)==1)return 1;
259     else if(mouse_press(110,358,110+24*6+6,358+24)==2)return 2;
260     else return 0;
261 }
262
263 int intrack5(void){
264     if(MouseX>(110-3)&&MouseY>414&&MouseX<(110+24*6+3)&&MouseY<(414+24))return
1;
265     else return 0;
266 }
267 int presstrack5(void){
268     if(mouse_press(110,414,110+24*6+6,414+24)==1)return 1;
269     else if(mouse_press(110,414,110+24*6+6,414+24)==2)return 2;
270     else return 0;
271 }
272
273 int intrack6(void){
274     if(MouseX>(380-3)&&MouseY>190&&MouseX<(380+24*6+3)&&MouseY<(190+24))return
1;
275     else return 0;

```

```
276 }
277 int presstrack6(void){
278     if(mouse_press(380,190,380+24*6+6,190+24)==1)return 1;
279     else if(mouse_press(380,190,380+24*6+6,190+24)==2)return 2;
280     else return 0;
281 }
282
283 int intrack7(void){
284     if(MouseX>(380-3)&&MouseY>246&&MouseX<(380+24*6+3)&&MouseY<(246+24))return
1;
285     else return 0;
286 }
287 int presstrack7(void){
288     if(mouse_press(380,246,380+24*6+6,246+24)==1)return 1;
289     else if(mouse_press(380,246,380+24*6+6,246+24)==2)return 2;
290     else return 0;
291 }
292
293 int intrack8(void){
294     if(MouseX>(380-3)&&MouseY>302&&MouseX<(380+24*6+3)&&MouseY<(302+24))return
1;
295     else return 0;
296 }
297 int presstrack8(void){
298     if(mouse_press(380,302,380+24*6+6,302+24)==1)return 1;
299     else if(mouse_press(380,302,380+24*6+6,302+24)==2)return 2;
300     else return 0;
301 }
302
303 int intrack9(void){
304     if(MouseX>(380-3)&&MouseY>358&&MouseX<(380+24*6+3)&&MouseY<(358+24))return
1;
305     else return 0;
306 }
307 int presstrack9(void){
308     if(mouse_press(380,358,380+24*6+6,358+24)==1)return 1;
309     else if(mouse_press(380,358,380+24*6+6,358+24)==2)return 2;
310     else return 0;
311 }
312
313 int intrack10(void){
314     if(MouseX>(380-3)&&MouseY>414&&MouseX<(380+24*6+3)&&MouseY<(414+24))return
1;
315     else return 0;
316 }
317 int presstrack10(void){
318     if(mouse_press(380,414,380+24*6+6,414+24)==1)return 1;
319     else if(mouse_press(380,414,380+24*6+6,414+24)==2)return 2;
```

```
320     else return 0;
321 }
```

4. drawpage.c

C

```
1  #include"public.h"
2
3  void drawstart(){
4      cleardevice();
5      setbkcolor(LIGHTGRAY);
6      drawtop();
7
8      barword32(0,90,BLUE,7,7,"请选择登录方式",LIGHTGRAY);
9      barword32(0,200,BLUE,5,4,"乘客登录",LIGHTGRAY);
10     barword32(0,300,BLUE,5,5,"管理员登录",LIGHTGRAY);
11     return;
12 }
13
14 void drawadlogin(){
15     cleardevice();
16     setbkcolor(LIGHTGRAY);
17     drawtop();
18
19     barword32(0,85,BLUE,6,5,"管理员登录",LIGHTGRAY);
20     barword32(60,170,BLUE,3,2,"账号",LIGHTGRAY);
21     barword32(60,250,BLUE,3,2,"密码",LIGHTGRAY);
22     barwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED);
23     barwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED);
24     barword32(0,350,BLUE,3,2,"登录",LIGHTGRAY);
25     returnbutton(LIGHTGRAY);
26
27     return;
28 }
29
30 void drawpallogin(){
31     cleardevice();
32     setbkcolor(LIGHTGRAY);
33     drawtop();
34
35     barword32(0,85,BLUE,5,4,"乘客登录",LIGHTGRAY);
36     barword32(60,170,BLUE,3,2,"账号",LIGHTGRAY);
37     barword32(60,250,BLUE,3,2,"密码",LIGHTGRAY);
38     barwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED);
39     barwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED);
```

```

40     barword32(0,350,BLUE,3,2,"登录",LIGHTGRAY);
41     barword16(450,420,BLUE,5,5,"新用户注册",LIGHTGRAY);
42     returnbutton(LIGHTGRAY);
43     return;
44 }
45
46 void drawparegister(){
47     cleardevice();
48     setbkcolor(LIGHTGRAY);
49     drawtop();
50     returnbutton(LIGHTGRAY);
51
52     barword32(0,85,BLUE,5,4,"乘客注册",LIGHTGRAY);
53     barword32(60,170,BLUE,3,2,"账号",LIGHTGRAY);
54     barwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED);
55
56     barword32(60,250,BLUE,3,2,"密码",LIGHTGRAY);
57     barwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED);
58
59     barwordframe(180,330,WHITE,11,"",LIGHTGRAY,RED);
60     puthz(190,335,"再次输入密码",32,32,LIGHTGRAY);
61     barword32(0,400,BLUE,2,2,"注册",LIGHTGRAY);
62 }
63
64 void drawpabind(){
65     cleardevice();
66     setbkcolor(LIGHTGRAY);
67     drawtop();
68     returnbutton(LIGHTGRAY);
69
70     barword32(0,120,BLUE,6,6,"绑定个人信息",WHITE);
71     puthz(340-10.5*16,70,"该账号尚未绑定个人信息，请绑定个人信息",16,16,RED);
72
73     barword32(60,200,BLUE,4,4,"身份证号",WHITE);
74     barwordframe(210,200,WHITE,11,"",LIGHTGRAY,RED);
75     barword32(60,300,BLUE,4,4,"电话号码",WHITE);
76     barwordframe(210,300,WHITE,11,"",LIGHTGRAY,RED);
77
78     barword32(0,400,BLUE,2,2,"绑定",WHITE);
79 }
80
81 void drawpamessage(struct passenger *pa){
82     char cname[10];
83
84     cleardevice();
85     setbkcolor(LIGHTGRAY);
86     drawtop();
87     returnbutton(LIGHTGRAY);

```

```

88
89     barword32(0,70,RED,6,4,"乘客信息",WHITE);
90     setfillstyle(1,WHITE);
91     bar(50,140,590,352);
92
93     puthz(70,150,"姓名",32,32,LIGHTGRAY);
94     puthz(250,150,"性别",32,32,LIGHTGRAY);
95     puthz(420,150,"年龄",32,32,LIGHTGRAY);
96     puthz(70,200,"身份证号",32,32,LIGHTGRAY);
97     puthz(70,250,"电话",32,32,LIGHTGRAY);
98     puthz(70,300,"健康状况",32,32,LIGHTGRAY);
99
100    barword32(65,400,BLUE,4,4,"登记乘车",LIGHTGRAY);
101    barword32(265,400,BLUE,4,4,"乘车记录",LIGHTGRAY);
102    barword32(465,400,BLUE,4,4,"轨迹查询",LIGHTGRAY);
103
104    getcnamebyID(pa->ID,cname);
105    puthz(140,150,cname,32,32,LIGHTGRAY);
106
107    if(pa->sex==1){
108        puthz(350,150,"男",32,32,LIGHTGRAY);
109    }
110    else if(pa->sex==2){
111        puthz(350,150,"女",32,32,LIGHTGRAY);
112    }
113    settextstyle(TRIPLEX_FONT,HORIZ_DIR,4);
114    putnum(515,145,pa->age);
115    outtextxy(205,195,pa->ID);
116    outtextxy(200,245,pa->tel);
117    if(pa->status==0){
118        puthz(300,300,"无异常",32,32,GREEN);
119    }
120    if(pa->status==1){
121        puthz(300,300,"新冠肺炎患者",32,32,RED);
122    }
123    if(pa->status==2){
124        puthz(300,300,"无症状感染者",32,32,RED);
125    }
126    if(pa->status==3){
127        puthz(300,300,"密切接触者",32,32,YELLOW);
128    }
129 }
130
131 void drawpapostrecord(){
132     cleardevice();
133     setbkcolor(LIGHTGRAY);
134     drawtop();
135     returnbutton(LIGHTGRAY);

```

```

135     returnbutton(LIGHTGRAY);
136
137     barword32(0,70,BLUE,4,4,"登记乘车",RED);
138     todaydate();
139
140     barword32(110,185,WHITE,2,2,"车次",LIGHTGRAY);
141     setcolor(BLUE);
142     settextstyle(1,0,5);
143     outtextxy(240,170,"G");
144     barwordframe(280,185,WHITE,3,"",0,RED);
145     barword32(430,185,BLUE,2,2,"确认",WHITE);
146 }
147
148 void drawpagetrecord(struct pagetrain *ptrnu,struct pagetrain *ptrnm,struct pa
getrain *ptrnl,int currentpage,int countpage){
149     cleardevice();
150     setbkcolor(LIGHTGRAY);
151     drawtop();
152     returnbutton(LIGHTGRAY);
153
154     barword32(0,65,WHITE,4,4,"乘车记录",RED);
155     if(ptrnu!=NULL)trlabelu(ptrnu);
156     if(ptrnm!=NULL)trlabelm(ptrnm);
157     if(ptrnl!=NULL)trlabell(ptrnl);
158
159     leftarrow(BLUE);
160     rightarrow(BLUE);
161     pagenumber(currentpage,countpage);
162
163     return ;
164 }
165
166 void drawpagetrack(){
167     cleardevice();
168     setbkcolor(LIGHTGRAY);
169     drawtop();
170     returnbutton(LIGHTGRAY);
171
172     drawmap();
173 }
174
175 void drawadlabel(){
176     cleardevice();
177     setbkcolor(LIGHTGRAY);
178     drawtop();
179     returnbutton(LIGHTGRAY);
180
181     puthz(320-32*3.5,80,"请录入阳性患者",32,32,GREEN);

```



```

182     barword32(80,150, GREEN,4,4,"身份证号",WHITE);
183     barwordframe(230,150,WHITE,11,"",0,RED);
184
185     barword24(80,240, GREEN,8,8,"是否无症状感染者",WHITE);
186     puthz(330,240,"是",24,24, GREEN);
187     puthz(430,240,"否",24,24, GREEN);
188     setcolor(GREEN);
189     circle(380,252,10);
190     circle(480,252,10);
191
192     barword32(0,320, GREEN,4,4,"确认录入",WHITE);
193     return;
194 }
195
196 void drawregister(){          //乘客登记信息界面
197
198     cleardevice();
199     setbkcolor(LIGHTGRAY);
200     drawtop();
201
202     barword32(75,150,BLUE,4,2,"姓名",WHITE);
203     barwordframe(250,150,WHITE,10,"",0,RED);
204     barword32(75,200,BLUE,4,4,"身份证号",WHITE);
205     barwordframe(250,200,WHITE,10,"",0,RED);
206     barword32(75,300,BLUE,4,3,"电话",WHITE);
207     barwordframe(250,300,WHITE,10,"",0,RED);
208     barword32(0,400,BLUE,2,2,"注册",WHITE);
209
210     return;
211 }
212
213 void drawstationselect(char *trainname){
214     cleardevice();
215     setbkcolor(LIGHTGRAY);
216     drawtop();
217     returnbutton(LIGHTGRAY);
218
219     barword32(0,60,BLUE,4,4,"车次信息",WHITE);
220     puthz(360,130,"请选择出发地与目的地",24,24,RED);
221     puthz(60,130,"列车号",32,32,RED);
222     setcolor(RED);
223     settextstyle(1,0,3);
224     outtextxy(200,130,trainname);
225     track1(trainname,WHITE);
226     track2(trainname,WHITE);
227     track3(trainname,WHITE);
228     track4(trainname,WHITE);
229     track5(trainname,WHITE);

```

```

230     track6(trainname,WHITE);
231     track7(trainname,WHITE);
232     track8(trainname,WHITE);
233     track9(trainname,WHITE);
234     track10(trainname,WHITE);
235 }
236
237 void drawadmanager(){
238     cleardevice();
239     setbkcolor(LIGHTGRAY);
240     drawtop();
241     returnbutton(LIGHTGRAY);
242
243     searchbutton(WHITE);
244     barwordframe(100,100,WHITE,11,"",WHITE,RED);
245     barword32(100,200,CYAN,6,6,"标记乘客状况",WHITE);
246     barword32(350,200,DARKGRAY,6,6,"查询阳性乘客",WHITE);
247     barword32(350,300,BLUE,6,6,"查询密接乘客",WHITE);
248     barword32(100,300,BROWN,6,6,"查询可疑车次",WHITE);
249 }
250
251 void drawadhealth(){
252     cleardevice();
253     setbkcolor(LIGHTGRAY);
254     drawtop();
255
256     puthz(25,150,"请输入身份证号",32,32,WHITE);
257     barwordframe(250,150,WHITE,11,"",0,RED);
258     barword32(320-32*6-10,300,BLUE,6,6,"查询健康状况",LIGHTGRAY);
259     barword32(320+10,300,RED,6,6,"查询出行记录",LIGHTGRAY);
260 }
261
262 void drawadgetpospa(struct passenger *pau,struct passenger *pam,struct passenger *pal,int currentpage,int countpage){
263     cleardevice();
264     setbkcolor(LIGHTGRAY);
265     drawtop();
266     returnbutton(LIGHTGRAY);
267
268     barword32(0,65,WHITE,6,6,"阳性乘客信息",RED);
269     if(pau!=NULL)palabelu(pau);
270     if(pam!=NULL)palabelm(pam);
271     if(pal!=NULL)palabell(pal);
272
273     leftarrow(BLUE);
274     rightarrow(BLUE);
275     pagenumber(currentpage,countpage);
276

```

```
276
277     return ;
278 }
279
280 void drawadgetcttpa(struct passenger *pau,struct passenger *pam,struct passenger *pal,int currentpage,int countpage){
281     cleardevice();
282     setbkcolor(LIGHTGRAY);
283     drawtop();
284     returnbutton(LIGHTGRAY);
285
286     barword32(0,65,WHITE,6,6,"密接乘客信息",RED);
287     if(pau!=NULL)palabelu(pau);
288     if(pam!=NULL)palabelm(pam);
289     if(pal!=NULL)palabell(pal);
290
291     leftarrow(BLUE);
292     rightarrow(BLUE);
293     pagenumber(currentpage,countpage);
294
295     return ;
296 }
297
298 void drawadgetpostr(struct pagetrain *ptrnu,struct pagetrain *ptrnm,struct pagetrain *ptrnl,int currentpage,int countpage){
299     cleardevice();
300     setbkcolor(LIGHTGRAY);
301     drawtop();
302     returnbutton(LIGHTGRAY);
303
304     barword32(0,65,WHITE,6,6,"阳性车次信息",RED);
305     if(ptrnu!=NULL)trlabelu(ptrnu);
306     if(ptrnm!=NULL)trlabelm(ptrnm);
307     if(ptrnl!=NULL)trlabell(ptrnl);
308
309     leftarrow(BLUE);
310     rightarrow(BLUE);
311     pagenumber(currentpage,countpage);
312 }
313
314 void drawadsearch(struct passenger *pau,struct passenger *pam,struct passenger *pal,int currentpage,int countpage){
315     cleardevice();
316     setbkcolor(LIGHTGRAY);
317     drawtop();
318     returnbutton(LIGHTGRAY);
319
320     barword32(0,65,WHITE,4,4,"搜索结果",RED);
```

```

321     if(pau!=NULL)palabelu(pau);
322     if(pam!=NULL)palabelm(pam);
323     if(pal!=NULL)palabell(pal);
324
325     leftarrow(BLUE);
326     rightarrow(BLUE);
327     pagenumber(currentpage,countpage);
328 }
329
330 void drawadpamessage(struct passenger *pa){
331     char cname[10];
332
333     cleardevice();
334     setbkcolor(LIGHTGRAY);
335     drawtop();
336     returnbutton(LIGHTGRAY);
337
338     barword32(0,70,RED,6,4,"乘客信息",WHITE);
339     setfillstyle(1,WHITE);
340     bar(50,140,590,352);
341
342     puthz(70,150,"姓名",32,32,LIGHTGRAY);
343     puthz(250,150,"性别",32,32,LIGHTGRAY);
344     puthz(420,150,"年龄",32,32,LIGHTGRAY);
345     puthz(70,200,"身份证号",32,32,LIGHTGRAY);
346     puthz(70,250,"电话",32,32,LIGHTGRAY);
347     puthz(70,300,"健康状况",32,32,LIGHTGRAY);
348
349     barword32(65,400,BLUE,4,4,"登记乘车",RED);
350     barword32(265,400,BLUE,4,4,"乘车记录",LIGHTGRAY);
351     barword32(465,400,BLUE,4,4,"轨迹查询",LIGHTGRAY);
352
353     getcnamebyID(pa->ID,cname);
354     puthz(140,150,cname,32,32,LIGHTGRAY);
355
356     if(pa->sex==1){
357         puthz(350,150,"男",32,32,LIGHTGRAY);
358     }
359     else if(pa->sex==2){
360         puthz(350,150,"女",32,32,LIGHTGRAY);
361     }
362     settextstyle(TRIPLEX_FONT,HORIZ_DIR,4);
363     putnum(515,145,pa->age);
364     outtextxy(205,195,pa->ID);
365     outtextxy(200,245,pa->tel);
366     if(pa->status==0){
367         puthz(300,300,"无异常",32,32,GREEN);
368     }

```

```

369     if(pa->status==1){
370         puthz(300,300,"新冠肺炎患者",32,32,RED);
371     }
372     if(pa->status==2){
373         puthz(300,300,"无症状感染者",32,32,RED);
374     }
375     if(pa->status==3){
376         puthz(300,300,"密切接触者",32,32,YELLOW);
377     }
378 }

```

5. file.c

C

```

1  #include"public.h"
2
3  void initfile(void){
4      FILE *log,*fadministrator,*fuser,*ftrain,*frecord,*fpassenger;
5
6      log=fopen(".\\LOGGER","w+");
7
8      fadministrator=fopen(".\\DB\\ADMIN","wb+");
9      fuser=fopen(".\\DB\\USER","wb+");
10     ftrain=fopen(".\\DB\\TRAIN","wb+");
11     frecord=fopen(".\\DB\\RECORD","wb+");
12     fpassenger=fopen(".\\DB\\PASSENG","wb+");
13
14     fclose(log);
15     fclose(fadministrator);
16     fclose(fuser);
17     fclose(ftrain);
18     fclose(frecord);
19     fclose(fpassenger);
20
21     writeuser("xyd","993995");
22     writeuser("zzl","993995");
23     writeadmin("xyd","993995");
24     writeadmin("zzl","993995");
25     writetrain("G562",2022,4,17);
26     writetrain("G186",2022,4,17);
27     writetrain("G379",2022,4,17);
28     writetrain("G335",2022,4,17);
29     writepassenger(1,"320602200308222547","15062744586",2,18,1);
30     writepassenger(2,"320602200211035913","18921673386",1,19,0);
31     writerecordv1(2,2,"fne");

```

```

32     writerecordv1(2,3,"fcd");
33     writerecordv1(2,4,"nkl");
34 }
35
36 int writeuser(char *name,char *password){
37     FILE* fuser;
38     FILE* log;
39     struct user* usr;
40     int n;
41
42     log=fopen(".\\LOGGER","a+");
43
44     if((fuser=fopen(".\\DB\\USER", "rb+" ))==NULL)
45     {
46         fprintf(log,"\nerr:open user fail!");
47         delay(3000);
48         exit(1);
49
50         fclose(log);
51         return 0;
52     }
53
54     fseek(fuser,0,SEEK_END);
55     n=ftell(fuser)/sizeof(USER);
56
57     if((usr=(USER*)malloc(sizeof(USER)))==NULL){
58         fprintf(log,"\nerr:no space for user!");
59         delay(3000);
60         exit(1);
61
62         fclose(log);
63         return 0;
64     }
65
66     usr->pakey=n+1;
67     strcpy(usr->name,name);
68     strcpy(usr->password,password);
69
70     fseek(fuser,0,SEEK_END);
71     fwrite(usr,sizeof(USER),1,fuser);
72     fflush(fuser);
73
74     if (usr!=NULL)
75     {
76         free(usr);
77         usr=NULL;
78     }
79

```

```

80     if (fclose(fuser)!=0)
81     {
82         fprintf(log, "\nerr:close user fail!");
83         delay(3000);
84         exit(1);
85
86         fclose(log);
87         return 0;
88     }
89
90     fprintf(log, "\ninfo:write user pakey %d name %s password %s success!", n+1,
name,password);
91
92     fclose(log);
93     return 1;
94 }
95
96 //pakey:登录成功 0:文件操作错误 -1: 密码错误 -2: 用户未注册 -3: 未知错误
97 int loginuser(char *name, char *password){
98     FILE* user;
99     FILE* log;
100     struct user* usr;
101     int i;
102     int n;
103     int pakey;
104
105     log=fopen(".\\LOGGER", "a+");
106
107     if((user=fopen(".\\DB\\USER", "rb+ "))==NULL)
108     {
109         fprintf(log, "\nerr:open user fail!");
110         delay(3000);
111         exit(1);
112         fclose(log);
113         return 0;
114     }
115     fseek(user, 0, SEEK_END);
116     n=ftell(user)/sizeof(USER);
117
118     for(i=0; i<n; i++){
119         if ((usr=(USER*)malloc(sizeof(USER)))==NULL){
120             fprintf(log, "\nerr:no space for user!");
121             delay(3000);
122             exit(1);
123
124             fclose(log);
125             return 0;

```

```
126     }
127     fseek(user,i*sizeof(USER),SEEK_SET);
128     fread(usr,sizeof(USER),1,user);
129
130     if(strcmp(name,usr->name)==0){
131         if(strcmp(password,usr->password)==0){
132             pakey=usr->pakey;
133
134             if (usr!=NULL){
135                 free(usr);
136                 usr=NULL;
137             }
138             if (fclose(user)!= 0){
139                 fprintf(log,"\nerr:close user fail!");
140                 delay(3000);
141                 exit(1);
142
143                 fclose(log);
144                 return 0;
145             }
146
147             fprintf(log,"\ninfo:user %s login success!",name);
148
149             fclose(log);
150             return pakey;
151
152         }
153         else if(strcmp(password,usr->password)!=0){
154             if (usr!=NULL){
155                 free(usr);
156                 usr=NULL;
157             }
158             if (fclose(user)!= 0){
159                 fprintf(log,"\nerr:close user fail!");
160                 delay(3000);
161                 exit(1);
162
163                 fclose(log);
164                 return 0;
165             }
166             fprintf(log,"\ninfo:user %s login fail:password wrong!",name);
167
168             fclose(log);
169             return -1;
170         }
171     }
172     if (usr!=NULL){
173         free(usr);
```



```

174         usr=NULL;
175     }
176
177 }
178 if(i==n){
179     if (usr!=NULL){
180         free(usr);
181         usr=NULL;
182     }
183     if (fclose(user)!= 0){
184         fprintf(log,"\nerr:close user fail!");
185         delay(3000);
186         exit(1);
187         fclose(log);
188         return 0;
189     }
190     fprintf(log,"\ninfo:user %s login fail:no register!",name);
191
192     fclose(log);
193     return -2;
194 }
195
196 fprintf(log,"\nerr:user %s login unknown fail!",name);
197
198 fclose(log);
199 return -3;
200 }
201
202 //1: 存在 -1: 不存在
203 int existusername(char *name){
204     FILE* fuser;
205     FILE* log;
206     struct user* usr;
207     int i;
208     int n;
209     log=fopen(".\\LOGGER","a+");
210
211     if((fuser=fopen(".\\DB\\USER", "rb+"))==NULL){
212         fprintf(log,"\nerr:open user fail!");
213         delay(3000);
214         exit(1);
215
216         fclose(log);
217         return 0;
218     }
219
220     fseek(fuser,0,SEEK_END);
221     n=ftell(fuser)/sizeof(USER);

```

```

221     if (fseek(fuser, 0, SEEK_SET) != 0) {
222
223         for(i=0; i<n; i++){
224             if((usr=(USER*)malloc(sizeof(USER)))==NULL){
225                 fprintf(log, "\nerr:no space for user!");
226                 delay(3000);
227                 exit(1);
228
229                 fclose(log);
230                 return 0;
231             }
232
233             fseek(fuser, i*sizeof(USER), SEEK_SET);
234             fread(usr, sizeof(USER), 1, fuser);
235             if(strcmp(name, usr->name)==0){
236                 if(usr != NULL){
237                     free(usr);
238                     usr = NULL;
239                 }
240                 if(fclose(fuser) != 0){
241                     printf(log, "\nerr:close user fail!");
242                     delay(3000);
243                     exit(1);
244
245                     fclose(log);
246                     return 0;
247                 }
248
249                 if(usr!= NULL){
250                     free(usr);
251                     usr = NULL;
252                 }
253                 fclose(log);
254                 return 1;
255             }
256             if(usr!= NULL){
257                 free(usr);
258                 usr = NULL;
259             }
260         }
261
262         if(fclose(fuser)!= 0){
263             printf(log, "\nerr:close user fail!");
264             delay(3000);
265             exit(1);
266
267             fclose(log);
268             return 0;

```

```

269     }
270
271     fclose(log);
272     return -1;
273 }
274
275 int writeadmin(char *name, char *password){
276     FILE* fadministrator;
277     FILE* log;
278     struct administrator* admin;
279     int n;
280
281     log=fopen(".\\LOGGER", "a+");
282
283     if((fadministrator=fopen(".\\DB\\ADMIN", "rb+" ))==NULL)
284     {
285         fprintf(log, "\\nerr:open administrator fail!");
286         delay(3000);
287         exit(1);
288
289         fclose(log);
290         return 0;
291     }
292
293     fseek(fadministrator, 0, SEEK_END);
294     n=ftell(fadministrator)/sizeof(ADMINISTRATOR);
295
296     if((admin=(ADMINISTRATOR*)malloc(sizeof(ADMINISTRATOR)))==NULL){
297         fprintf(log, "\\nerr:no space for administrator!");
298         delay(3000);
299         exit(1);
300
301         fclose(log);
302         return 0;
303     }
304
305     admin->adkey=n+1;
306     strcpy(admin->name, name);
307     strcpy(admin->password, password);
308
309     fseek(fadministrator, 0, SEEK_END);
310     fwrite(admin, sizeof(ADMINISTRATOR), 1, fadministrator);
311     fflush(fadministrator);
312
313     if (admin!=NULL)
314     {
315         free(admin);
316         admin=NULL;

```

```

317     }
318
319     if (fclose(fadministrator)!=0)
320     {
321         fprintf(log, "\nerr:close administrator fail!");
322         delay(3000);
323         exit(1);
324
325         fclose(log);
326         return 0;
327     }
328
329     fprintf(log, "\ninfo:write administrator adkey %d name %s password %s succe
ss!", n+1, name, password);
330
331     fclose(log);
332     return 1;
333 }
334
335 //1:登录成功 0:文件操作错误 -1: 密码错误 -2: 用户未注册 -3: 未知错误
336 int loginadmin(char *name, char *password){
337     FILE* fadministrator;
338     FILE* log;
339     struct administrator* admin;
340     int i;
341     int n;
342
343     log=fopen(".\\LOGGER", "a+");
344
345     if((fadministrator=fopen(".\\DB\\ADMIN", "rb+" ))==NULL)
346     {
347         fprintf(log, "\nerr:open administrator fail!");
348         delay(3000);
349         exit(1);
350
351         fclose(log);
352         return 0;
353     }
354     fseek(fadministrator, 0, SEEK_END);
355
356     n=ftell(fadministrator)/sizeof(ADMINISTRATOR);
357
358     for(i=0; i<n; i++){
359         if ((admin=(ADMINISTRATOR*)malloc(sizeof(ADMINISTRATOR)))==NULL){
360             fprintf(log, "\nerr:no space for administrator!");
361             delay(3000);
362             exit(1);
363         }

```

```

362
363         fclose(log);
364         return 0;
365     }
366     fseek(fadministrator,i*sizeof(ADMINISTRATOR),SEEK_SET);
367     fread(admin,sizeof(ADMINISTRATOR),1,fadministrator);
368
369     if(strcmp(name,admin->name)==0){
370         if(strcmp(password,admin->password)==0){
371             if (admin!=NULL){
372                 free(admin);
373                 admin=NULL;
374             }
375             if (fclose(fadministrator)!= 0){
376                 fprintf(log,"\nerr:close administrator fail!");
377                 delay(3000);
378                 exit(1);
379
380                 fclose(log);
381                 return 0;
382             }
383
384             fprintf(log,"\ninfo:adminnistrator %s login success!",name);
385
386             fclose(log);
387             return 1;
388         }
389     }
390     else if(strcmp(password,admin->password)!=0){
391         if (admin!=NULL){
392             free(admin);
393             admin=NULL;
394         }
395         if (fclose(fadministrator)!= 0){
396             fprintf(log,"\nerr:close administrator fail!");
397             delay(3000);
398             exit(1);
399
400             fclose(log);
401             return 0;
402         }
403         fprintf(log,"\ninfo:administrator %s login fail:password wron
404 g!",name);
405
406         fclose(log);
407         return -1;
408     }

```

```

409         if (admin!=NULL){
410             free(admin);
411             admin=NULL;
412         }
413     }
414     if(i==n){
415         if (admin!=NULL){
416             free(admin);
417             admin=NULL;
418         }
419         if (fclose(fadministrator)!= 0){
420             fprintf(log,"\nerr:close administrator fail!");
421             delay(3000);
422             exit(1);
423             fclose(log);
424             return 0;
425         }
426         fprintf(log,"\ninfo:administrator %s login fail:no register!",name);
427
428         fclose(log);
429         return -2;
430     }
431
432     fprintf(log,"\nerr:administrator %s login unknown fail!",name);
433
434     fclose(log);
435     return -3;
436 }
437
438 int writepassenger(int pakey,char *ID,char *tel,int sex,int age,int status){
439     FILE* fpassenger;
440     FILE* log;
441     struct passenger *pa;
442
443     log=fopen(".\\LOGGER","a+");
444
445     if((fpassenger=fopen(".\\DB\\PASSENG", "rb+" ))==NULL){
446         fprintf(log,"\nerr:open passenger fail!");
447         delay(3000);
448         exit(1);
449
450         fclose(log);
451         return 0;
452     }
453
454     if((pa=(PASSENGER*)malloc(sizeof(PASSENGER)))==NULL){
455         fprintf(log,"\nerr:no space for passenger!");
456         delay(3000);

```

```

457         exit(1);
458
459         fclose(log);
460         return 0;
461     }
462
463     memset(pa, '\0', sizeof(pa));
464
465     pa->pakey=pakey;
466     pa->age=age;
467     pa->sex=sex;
468     pa->status=status;
469     strcpy(pa->ID, ID);
470     strcpy(pa->tel, tel);
471
472     fseek(fpasenger, 0, SEEK_END);
473     fwrite(pa, sizeof(PASSENGER), 1, fpasenger);
474     fflush(fpasenger);
475
476     if (pa != NULL){
477         free(pa);
478         pa = NULL;
479     }
480
481     if (fclose(fpasenger) != 0){
482         fprintf(log, "\nerr:close pasenger fail!");
483         delay(3000);
484         exit(1);
485
486         fclose(log);
487         return 0;
488     }
489
490     fprintf(log, "\ninfo:write pasenger pakey %d ID %s tel %s sex %d age %d st
atus %d success!", pakey, ID, tel, sex, age, status);
491     fclose(log);
492     return 1;
493 }
494
495 //1.存在 -1.不存在
496 int existpasengerID(char *ID){
497     FILE* fpasenger;
498     FILE* log;
499     struct passenger *pa;
500     int i;
501     int n;
502
503     log=fopen(" \\ LOGFILE \\ log.txt",

```

```

503     log=fopen(".\\LOGGER","a+");
504
505     if((fpassenger=fopen(".\\DB\\PASSENG","rb+"))==NULL){
506         fprintf(log,"\\nerr:open passenger fail!");
507         delay(3000);
508         exit(1);
509
510         fclose(log);
511         return 0;
512     }
513
514     fseek(fpassenger,0,SEEK_END);
515     n=ftell(fpassenger)/sizeof(PASSENGER);
516
517     for(i=0;i<n;i++){
518         if((pa=(PASSENGER*)malloc(sizeof(PASSENGER)))==NULL){
519             fprintf(log,"\\nerr:no space for passenger!");
520             delay(3000);
521             exit(1);
522
523             fclose(log);
524             return 0;
525         }
526
527         fseek(fpassenger,i*sizeof(PASSENGER),SEEK_SET);
528         fread(pa,sizeof(PASSENGER),1,fpassenger);
529         if(strcmp(ID,pa->ID)==0){
530             if(pa!= NULL){
531                 free(pa);
532                 pa= NULL;
533             }
534
535             if(fclose(fpassenger) != 0){
536                 printf(log,"\\nerr:close passenger fail!");
537                 delay(3000);
538                 exit(1);
539
540                 fclose(log);
541                 return 0;
542             }
543
544             fclose(log);
545             return 1;
546         }
547         if(pa!= NULL){
548             free(pa);
549             pa= NULL;
550         }

```



```

551     }
552
553     if(fclose(fpassenger) != 0){
554         printf(log, "\nerr:close passenger fail!");
555         delay(3000);
556         exit(1);
557
558         fclose(log);
559         return 0;
560     }
561
562     fclose(log);
563     return -1;
564 }
565
566 // -1: 没有该pakey 1: 成功获取
567 int getpassengerbypakey(int pakey, struct passenger *destpa){
568     FILE* fpassenger;
569     FILE* log;
570     struct passenger *pa;
571     int i;
572     int n;
573
574     log=fopen(".\\LOGGER", "a+");
575
576     if((fpassenger=fopen(".\\DB\\PASSENG", "rb+" ))==NULL){
577         fprintf(log, "\nerr:open passenger fail!");
578         fclose(log);
579
580         delay(3000);
581         exit(1);
582
583         return 0; //???
584     }
585
586     fseek(fpassenger, 0, SEEK_END);
587     n=ftell(fpassenger)/sizeof(PASSENGER);
588
589     for(i=0; i<n; i++){
590         if ((pa=(PASSENGER*)malloc(sizeof(PASSENGER)))==NULL){
591             fprintf(log, "\nerr:no space for passenger!");
592             fclose(log);
593
594             delay(3000);
595             exit(1);
596
597             return 0;
598         }

```

```

599
600     fseek(fpassenger, i*sizeof(PASSENGER), SEEK_SET);
601     fread(pa, sizeof(PASSENGER), 1, fpassenger);
602
603     if(pa->pakey==pakey){
604         destpa->pakey=pa->pakey;
605         destpa->age=pa->age;
606         destpa->status=pa->status;
607         destpa->sex=pa->sex;
608         strcpy(destpa->ID, pa->ID);
609         strcpy(destpa->tel, pa->tel);
610
611         if (pa!=NULL){
612             free(pa);
613             pa=NULL;
614         }
615
616         if (fclose(fpassenger) != 0){
617             fprintf(log, "\nerr:close passenger fail!");
618             fclose(log);
619
620             delay(3000);
621             exit(1);
622
623             return 0;
624         }
625
626         fprintf(log, "\ninfo:get passenger by pakey %d success!", pakey);
627
628         fclose(log);
629         return 1;
630     }
631
632     if (pa!=NULL){
633         free(pa);
634         pa=NULL;
635     }
636 }
637 if(i==n){
638     if (pa!=NULL){
639         free(pa);
640         pa=NULL;
641     }
642     if (fclose(fpassenger) != 0){
643         fprintf(log, "\nerr:close passenger fail!");
644         fclose(log);
645
646         delay(3000);

```

```

646         delay(3000);
647         exit(1);
648
649         return 0;
650     }
651
652     fprintf(log, "\ninfo:get passenger by pakey %d fail:no bind passenger
(no pakey in passenger)!", pakey);
653
654     fclose(log);
655     return -1;
656 }
657 }
658
659 //返回获得的个数
660 int getpassengerbystatus(int status, struct passenger destpa[20]){
661     FILE* fpassenger;
662     FILE* log;
663     struct passenger *pa;
664     int i;
665     int n;
666     int j=0;
667
668     log=fopen(".\\LOGGER", "a+");
669
670     if((fpassenger=fopen(".\\DB\\PASSENG", "rb+" ))==NULL){
671         printf("\nerr:open passenger fail!");
672         delay(3000);
673         exit(1);
674
675         fclose(log);
676         return 0;
677     }
678
679     fseek(fpassenger, 0, SEEK_END);
680     n=ftell(fpassenger)/sizeof(PASSENGER);
681
682     for(i=0; i<n; i++){
683         if ((pa=(PASSENGER*)malloc(sizeof(PASSENGER)))==NULL){
684             fprintf(log, "\nerr:no space for passenger!");
685             delay(3000);
686             exit(1);
687
688             fclose(log);
689             return 0;
690         }
691
692         fseek(fpassenger, i*sizeof(PASSENGER), SEEK_SET);

```

```

693         fread(pa,sizeof(PASSENGER),1,fpassenger);
694
695         if(pa->status==status|| (status==1&&pa->status==2)){
696             destpa[j].pakey=pa->pakey;
697             destpa[j].sex=pa->sex;
698             destpa[j].age=pa->age;
699             destpa[j].status=pa->status;
700             strcpy(destpa[j].ID,pa->ID);
701             strcpy(destpa[j].tel,pa->tel);
702
703             j++;
704             fprintf(log,"\ninfo:get passenger:pakey %d ID %s tel %s sex %d age
%d by status %d success!"
705                   ,pa->pakey,pa->ID,pa->tel,pa->sex,pa->age,pa->status);
706
707         }
708         if (pa!=NULL){
709             free(pa);
710             pa=NULL;
711         }
712
713         if (fclose(fpassenger)!= 0){
714             fprintf(log,"\nerr:close passenger fail!");
715             delay(3000);
716             exit(1);
717
718             fclose(log);
719             return 0;
720         }
721
722         fclose(log);
723         return j;
724     }
725
726     int getpassengerbysearch(char *searchID,struct passenger destpa[20]){
727         FILE* fpassenger;
728         FILE* log;
729         struct passenger *pa;
730         int i;
731         int n;
732         int j=0;
733
734         log=fopen(".\\LOGGER","a+");
735
736         if((fpassenger=fopen(".\\DB\\PASSENG", "rb+" ))==NULL){
737             printf("\nerr:open passenger fail!");
738             delay(3000);

```

```

738     delay(3000);
739     exit(1);
740
741     fclose(log);
742     return 0; //???
743 }
744
745 fseek(fpasenger,0,SEEK_END);
746 n=ftell(fpasenger)/sizeof(PASSENGER);
747
748 for(i=0;i<n;i++){
749     if ((pa=(PASSENGER*)malloc(sizeof(PASSENGER)))==NULL){
750         fprintf(log,"\nerr:no space for passenger!");
751         delay(3000);
752         exit(1);
753
754         fclose(log);
755         return 0;
756     }
757
758     fseek(fpasenger,i*sizeof(PASSENGER),SEEK_SET);
759     fread(pa,sizeof(PASSENGER),1,fpasenger);
760
761     if(samestringmax(searchID,pa->ID)>=6){
762         destpa[j].pakey=pa->pakey;
763         destpa[j].sex=pa->sex;
764         destpa[j].age=pa->age;
765         destpa[j].status=pa->status;
766         strcpy(destpa[j].ID,pa->ID);
767         strcpy(destpa[j].tel,pa->tel);
768
769         j++;
770         fprintf(log,"\ninfo:get passenger:pakey %d ID %s tel %s sex %d age
%d by search %s success!"
,pa->pakey,pa->ID,pa->tel,pa->sex,pa->age,searchID);
771     }
772 }
773 if (pa!=NULL){
774     free(pa);
775     pa=NULL;
776 }
777 }
778
779 if (fclose(fpasenger)!= 0){
780     fprintf(log,"\nerr:close passenger fail!");
781     delay(3000);
782     exit(1);
783
784     fclose(log);

```

```

785         return 0;
786     }
787
788     fclose(log);
789     return j;
790 }
791
792 // -1: 没有相应的ID 1: 操作成功
793 int getpakeybyID(char *ID, int *pakey){
794     FILE* fpassenger;
795     FILE* log;
796     struct passenger *pa;
797     int i;
798     int n;
799
800     log=fopen(".\\LOGGER", "a+");
801
802     if((fpassenger=fopen(".\\DB\\PASSENG", "rb+" ))==NULL){
803         fprintf(log, "\nerr:open passenger fail!");
804         delay(3000);
805         exit(1);
806
807         fclose(log);
808         return 0;
809     }
810
811     fseek(fpassenger, 0, SEEK_END);
812     n=ftell(fpassenger)/sizeof(PASSENGER);
813
814     for(i=0; i<n; i++){
815         if ((pa=(PASSENGER*)malloc(sizeof(PASSENGER)))==NULL){
816             fprintf(log, "\nerr:no space for passenger!");
817             delay(3000);
818             exit(1);
819
820             fclose(log);
821
822             return 0;
823         }
824         fseek(fpassenger, i*sizeof(PASSENGER), SEEK_SET);
825         fread(pa, sizeof(PASSENGER), 1, fpassenger);
826         // fprintf(log, "\ndebug:%s", pa->ID); //ok
827         if(strcmp(pa->ID, ID)==0){
828
829             *pakey=pa->pakey;
830
831             if (pa!=NULL){
832                 free(pa);

```

```

833         pa=NULL;
834
835     }
836
837     if (fclose(fpasenger) != 0){
838         fprintf(log, "\nerr:close passenger fail!");
839         delay(3000);
840         exit(1);
841
842         fclose(log);
843         return 0;
844     }
845
846     fprintf(log, "\ninfo:get pakey by ID %s success:pakey %d!", ID, *pake
y);
847     fclose(log);
848     return 1;
849 }
850 if (pa!=NULL){
851     free(pa);
852     pa=NULL;
853 }
854 }
855 if(i==n){
856     if (pa!=NULL){
857         free(pa);
858         pa=NULL;
859     }
860     if (fclose(fpasenger) != 0){
861         fprintf(log, "\nerr:close passenger fail!");
862         delay(3000);
863         exit(1);
864
865         fclose(log);
866         return 0;
867     }
868     fprintf(log, "\ninfo:get pakey by ID %s fail:no ID in passenger!", ID);
869     return -1;
870 }
871 }
872
873 // -2: 相同状态 -1: 没有pakey 1: 操作成功
874 int updatestatusbypakey(int pakey, int status){
875     FILE* fpasenger;
876     FILE* log;
877     struct passenger *pa;
878     int i;
879     int n;

```

```

879     int n,
880
881     log=fopen(".\\LOGGER","a+");
882
883     if((fpassenger=fopen(".\\DB\\PASSENG", "rb+" ))==NULL){
884         fprintf(log, "\\nerr:open passenger fail!");
885         delay(3000);
886         exit(1);
887
888         fclose(log);
889         return 0;
890     }
891
892     fseek(fpassenger,0,SEEK_END);
893     n=ftell(fpassenger)/sizeof(PASSENGER);
894
895     for(i=0;i<n;i++){
896         if ((pa=(PASSENGER*)malloc(sizeof(PASSENGER)))==NULL){
897             fprintf(log, "\\nerr:no space for user!");
898             delay(3000);
899             exit(1);
900
901             fclose(log);
902             return 0;
903         }
904         fseek(fpassenger,i*sizeof(PASSENGER),SEEK_SET);
905         fread(pa,sizeof(PASSENGER),1,fpassenger);
906
907
908         if(pa->pakey==pakey){
909             if(pa->status==status){
910                 fprintf(log, "\\ninfo:update status %d by pakey %d fail:same sta
911 tus %d!",status,pakey,status);
912
913                 if (pa!=NULL){
914                     free(pa);
915                     pa=NULL;
916                 }
917
918                 if (fclose(fpassenger)!= 0){
919                     fprintf(log, "\\nerr:close passenger fail!");
920                     delay(3000);
921                     exit(1);
922
923                     fclose(log);
924                     return 0;
925                 }

```



```

926         fclose(log);
927         return -2;
928     }
929
930     if(status==3&&(pa->status==1 || pa->status==2)){
931         fprintf(log, "\ninfo:update status %d by pakey %d fail:ctt to p
os!", status, pakey);
932
933         if (pa!=NULL){
934             free(pa);
935             pa=NULL;
936         }
937
938         if (fclose(fpasenger)!= 0){
939             fprintf(log, "\nerr:close passenger fail!");
940             delay(3000);
941             exit(1);
942
943             fclose(log);
944             return 0;
945         }
946
947         fclose(log);
948         return -2;
949     }
950
951     pa->status=status;
952     fseek(fpasenger, i*sizeof(PASSENGER), SEEK_SET);
953     fwrite(pa, sizeof(PASSENGER), 1, fpasenger);
954
955     if (fclose(fpasenger)!= 0){
956         fprintf(log, "\nerr:close passenger fail!");
957         delay(3000);
958         exit(1);
959
960         fclose(log);
961         return 0;
962     }
963
964     if (pa!=NULL){
965         free(pa);
966         pa=NULL;
967     }
968
969     fprintf(log, "\ninfo:update status %d by pakey %d success!", status,
pakey);
970     fclose(log);
971     return 1;

```

```

972     }
973     if (pa!=NULL){
974         free(pa);
975         pa=NULL;
976     }
977 }
978 if(i==n){
979     if (pa!=NULL){
980         free(pa);
981         pa=NULL;
982     }
983     if (fclose(fpasenger)!= 0){
984         fprintf(log,"\nerr:close passenger fail!");
985         delay(3000);
986         exit(1);
987
988         fclose(log);
989         return 0;
990     }
991     fprintf(log,"\ninfo:update status by pakey %d fail:no pakey in passeng
er!",pakey);
992     fclose(log);
993     return -1;
994 }
995 }
996
997 //-1: 日期错误
998 //fprintf
999 int writetrain(char *trainname,int yy,int mm,int dd){
1000     FILE* ftrain;
1001     FILE* log;
1002     struct train* trn;
1003     int n;
1004     int date;
1005
1006     trn=(struct train*)malloc(sizeof(TRAIN));
1007     log=fopen(".\\LOGGER","a+");
1008
1009     if((ftrain=fopen(".\\DB\\TRAIN", "rb+" ))==NULL){
1010         printf("\nerr:open train fail!");
1011         delay(3000);
1012         exit(1);
1013
1014         fclose(log);
1015         return 0;
1016     }
1017
1018     fclose(ftrain);

```

```

1018     tseek(ftrain,0,SEEK_END);
1019     n=ftell(ftrain)/sizeof(TRAIN);
1020
1021     if((trn=(TRAIN*)malloc(sizeof(TRAIN)))==NULL){
1022         printf("\nerr:no space for train!");
1023         delay(3000);
1024         exit(1);
1025         return 0;
1026     }
1027
1028     trn->trkey=n+1;
1029     strcpy(trn->trainname,trainname);
1030     //todo:加入判断trainname
1031     if(postdate(yy,mm,dd,&date)!=1){
1032         fprintf(log,"\ninfo:write train fail:wrong date!");
1033
1034         if (fclose(ftrain)!=0) {
1035             fprintf(log,"\nerr:close train fail!");
1036             delay(3000);
1037             exit(1);
1038
1039             fclose(log);
1040             return 0;
1041         }
1042
1043         fclose(log);
1044         free(trn);
1045         return -1;
1046     }
1047     //todo:?
1048     trn->date=date;
1049     trn->status=0;
1050
1051     fseek(ftrain,0,SEEK_END);
1052     fwrite(trn,sizeof(TRAIN),1,ftrain);
1053     fflush(ftrain); //todo
1054
1055     if (trn != NULL)
1056     {
1057         free(trn);
1058         trn = NULL;
1059     }
1060
1061     if (fclose(ftrain)!=0) //关闭文
1062     {
1063         fprintf(log,"\nerr:close train fail!");
1064         delay(3000);

```

```

1065         exit(1);
1066
1067         fclose(log);
1068         return 0;
1069     }
1070
1071     fprintf(log, "\ninfo:write train trkey %d trainname %s date %d.%d.%d success!",n+1,trainname,yy,mm,dd);
1072
1073     fclose(log);
1074     free(trn);
1075     return 1;
1076 }
1077
1078 // -1: 没有该trkey 1: 成功获取
1079 int gettrainbytrkey(int trkey, struct train *destrtn){
1080     FILE* ftrain;
1081     FILE* log;
1082     struct train *trn;
1083     int i;
1084     int n;
1085
1086     trn=(struct train *)malloc(sizeof(TRAIN));
1087     log=fopen(".\\LOGGER","a+");
1088
1089     if((ftrain=fopen(".\\DB\\TRAIN", "rb+" ))==NULL){
1090         fprintf(log, "\nerr:open train fail!");
1091         delay(3000);
1092         exit(1);
1093
1094         fclose(log);
1095         return 0; //???
1096     }
1097
1098     fseek(ftrain,0,SEEK_END);
1099     n=ftell(ftrain)/sizeof(TRAIN);
1100
1101     for(i=0;i<n;i++){
1102         if ((trn=(TRAIN*)malloc(sizeof(TRAIN)))==NULL){
1103             fprintf(log, "\nerr:no space for train!");
1104             delay(3000);
1105             exit(1);
1106
1107             fclose(log);
1108             return 0;
1109         }
1110
1111         fseek(ftrain,i*sizeof(TRAIN),SEEK_SET);

```

```

1112         fread(trn,sizeof(TRAIN),1,ftrain);
1113
1114         if(trn->trkey==trkey){
1115             desttrn->trkey=trn->trkey;
1116             desttrn->date=trn->date;
1117             desttrn->status=trn->status;
1118             strcpy(desttrn->trainname,trn->trainname);
1119
1120             if (trn!=NULL){
1121                 free(trn);
1122                 trn=NULL;
1123             }
1124
1125             if (fclose(ftrain)!= 0){
1126                 fprintf(log,"\nerr:close train fail!");
1127                 delay(3000);
1128                 exit(1);
1129
1130                 fclose(log);
1131                 return 0;
1132             }
1133
1134             fprintf(log,"\ninfo:get train by trkey %d success!",trkey);
1135
1136             fclose(log);
1137             free(trn);
1138             return 1;
1139         }
1140         if (trn!=NULL){
1141             free(trn);
1142             trn=NULL;
1143         }
1144     }
1145
1146     if(i==n){
1147         if (trn!=NULL){
1148             free(trn);
1149             trn=NULL;
1150         }
1151
1152         if (fclose(ftrain)!= 0){
1153             fprintf(log,"\nerr:close train fail!");
1154             delay(3000);
1155             exit(1);
1156
1157             fclose(log);
1158             return 0;

```

```

1159     }
1160
1161     fprintf(log, "\\ninfo:get train by trkey %d fail:no trkey in train!", trk
ey);
1162
1163     fclose(log);
1164     free(trn);
1165     return -1;
1166 }
1167 }
1168
1169 //返回获得的个数
1170 int gettrainbystatus(int status, struct pagetrain desttrn[20]){
1171     FILE* ftrain;
1172     FILE* log;
1173     struct train* trn;
1174     int i;
1175     int j=0;
1176     int n;
1177
1178     log=fopen(".\\\\LOGGER", "a+");
1179
1180     if((ftrain=fopen(".\\\\DB\\\\TRAIN", "rb+" ))==NULL){
1181         printf("\\nerr:open train fail!");
1182         delay(3000);
1183         exit(1);
1184
1185         fclose(log);
1186         return 0; //???
1187     }
1188
1189     fseek(ftrain, 0, SEEK_END);
1190     n=ftell(ftrain)/sizeof(TRAIN);
1191
1192     for(i=0; i<n; i++){
1193         if ((trn=(TRAIN*)malloc(sizeof(TRAIN)))==NULL){
1194             fprintf(log, "\\nerr:no space for train!");
1195             delay(3000);
1196             exit(1);
1197
1198             fclose(log);
1199             return 0;
1200         }
1201
1202         fseek(ftrain, i*sizeof(TRAIN), SEEK_SET);
1203         fread(trn, sizeof(TRAIN), 1, ftrain);
1204
1205         if(trn->status==status){

```

```

1206         desttrn[j].date=trn->date;
1207         desttrn[j].status=trn->status;
1208         desttrn[j].trkey=trn->trkey;
1209         strcpy(desttrn[j].trainname,trn->trainname);
1210
1211         j++;
1212         fprintf(log,"\ninfo:get train:trkey %d trainname %s date %d by sta
tus %d success!"
1213               ,trn->trkey,trn->trainname,trn->date,trn->status);
1214     }
1215     if (trn!=NULL){
1216         free(trn);
1217         trn=NULL;
1218     }
1219 }
1220
1221 if (fclose(ftrain)!= 0){
1222     fprintf(log,"\nerr:close train fail!");
1223     delay(3000);
1224     exit(1);
1225
1226     fclose(log);
1227     return 0;
1228 }
1229
1230 fclose(log);
1231 return j;
1232 }
1233
1234 int updatestatusbytrkey(int trkey,int status){
1235     FILE* ftrain;
1236     FILE* log;
1237     struct train *trn;
1238     int i;
1239     int n;
1240
1241     log=fopen(".\\LOGGER","a+");
1242
1243     if((ftrain=fopen(".\\DB\\TRAIN", "rb+" ))==NULL){
1244         fprintf(log,"\nerr:open train fail!");
1245         delay(3000);
1246         exit(1);
1247
1248         fclose(log);
1249         return 0;
1250     }
1251
1252     fseek(ftrain,0,SEEK_END);

```

```

1252     fseek(ftrain,0,SEEK_END);
1253     n=ftell(ftrain)/sizeof(TRAIN);
1254
1255     for(i=0;i<n;i++){
1256         if ((trn=(TRAIN*)malloc(sizeof(TRAIN)))==NULL){
1257             fprintf(log,"\nerr:no space for train!");
1258             delay(3000);
1259             exit(1);
1260
1261             fclose(log);
1262             return 0;
1263         }
1264         fseek(ftrain,i*sizeof(TRAIN),SEEK_SET);
1265         fread(trn,sizeof(TRAIN),1,ftrain);
1266
1267         if(trn->trkey==trkey){
1268             if(trn->status==status){
1269                 fprintf(log,"\ninfo:update status %d by trkey %d fail:same sta
1270 tus %d!",status,trkey,status);
1271
1272                 if (fclose(ftrain)!= 0){
1273                     fprintf(log,"\nerr:close user fail!");
1274                     delay(3000);
1275                     exit(1);
1276
1277                     fclose(log);
1278                     return 0;
1279                 }
1280
1281                 fclose(log);
1282                 return -2;
1283             }
1284
1285             trn->status=status;
1286
1287             fseek(ftrain,i*sizeof(TRAIN),SEEK_SET);
1288             fwrite(trn,sizeof(TRAIN),1,ftrain);
1289             fflush(ftrain);
1290
1291             if (trn!=NULL){
1292                 free(trn);
1293                 trn=NULL;
1294             }
1295             if (fclose(ftrain)!= 0){
1296                 fprintf(log,"\nerr:close user fail!");
1297                 delay(3000);
1298                 exit(1);

```



```

1299         fclose(log);
1300         return 0;
1301     }
1302
1303     fprintf(log, "\ninfo:update status %d by trkey %d success!", status,
trkey);
1304     fclose(log);
1305     return 1;
1306 }
1307 if (trn!=NULL){
1308     free(trn);
1309     trn=NULL;
1310 }
1311 }
1312
1313 if(i==n){
1314     if (trn!=NULL){
1315         free(trn);
1316         trn=NULL;
1317     }
1318     if (fclose(ftrain)!= 0){
1319         fprintf(log, "\nerr:close train fail!");
1320         delay(3000);
1321         exit(1);
1322
1323         fclose(log);
1324         return 0;
1325     }
1326     fprintf(log, "\ninfo:update status by trkey %d fail:otrkey in train!",
trkey);
1327
1328     fclose(log);
1329     return -1;
1330 }
1331 }
1332
1333 int getpostedtrkey(int date, char *trainname, int *trkey){
1334     FILE* ftrain;
1335     FILE* log;
1336     struct train *trn;
1337     int i;
1338     int n;
1339
1340     log=fopen(".\\LOGGER", "a+");
1341
1342     if((ftrain=fopen(".\\DB\\TRAIN", "rb+"))==NULL){
1343         fprintf(log, "\nerr:open train fail!");
1344         delay(3000);

```

```

1345         exit(1);
1346
1347         fclose(log);
1348         return 0;
1349     }
1350
1351     fseek(ftrain,0,SEEK_END);
1352     n=ftell(ftrain)/sizeof(TRAIN);
1353
1354     for(i=0;i<n;i++){
1355         if((trn=(TRAIN*)malloc(sizeof(TRAIN)))==NULL){
1356             fprintf(log,"\nerr:no space for train!");
1357             delay(3000);
1358             exit(1);
1359
1360             fclose(log);
1361             return 0;
1362         }
1363
1364         fseek(ftrain,i*sizeof(TRAIN),SEEK_SET);
1365         fread(trn,sizeof(TRAIN),1,ftrain);
1366         if((strcmp(trainname,trn->trainname)==0)&&(date==trn->date)){
1367             *trkey=trn->trkey;
1368
1369             if(trn!= NULL){
1370                 free(trn);
1371                 trn= NULL;
1372             }
1373
1374             if(fclose(ftrain) != 0){
1375                 printf(log,"\nerr:close train fail!");
1376                 delay(3000);
1377                 exit(1);
1378
1379                 fclose(log);
1380                 return 0;
1381             }
1382
1383             fprintf(log,"\ninfo:get trkey %d by trainname %s date %d success!"
, *trkey, trainname, date);
1384
1385             fclose(log);
1386             return 1;
1387         }
1388
1389         if(trn!= NULL){
1390             free(trn);
1391             trn= NULL;

```

```

1391         err= NULL,
1392     }
1393 }
1394
1395 if(fclose(ftrain)!= 0){
1396     printf(log,"\nerr:close train fail!");
1397     delay(3000);
1398     exit(1);
1399
1400     fclose(log);
1401     return 0;
1402 }
1403
1404 fprintf(log,"\ninfo:get trkey by trainname %s date %d fail!",trainname,date);
1405
1406 fclose(log);
1407 return -1;
1408 }

```

6. file2.c

C

```

1  #include"public.h"
2
3  //-1 ??
4  int writerecord(int pakey,int trkey){
5      FILE* frecord;
6      FILE* log;
7      struct record* rcd;
8      int i;
9      int n;
10
11     log=fopen("./\\LOGGER","a+");
12
13     if((frecored=fopen("./\\DB\\RECORD", "rb+"))==NULL){
14         fprintf(log,"\nerr:open record fail! %s",strerror(errno));
15         delay(3000);
16         exit(1);
17
18         fclose(log);
19         return 0;
20     }
21
22     fseek(frecord,0,SEEK_END);
23     n=ftell(frecord)/sizeof(RECORD);

```

```

24
25     for(i=0;i<n;i++){
26         if ((rcd=(RECORD*)malloc(sizeof(RECORD)))==NULL){
27             fprintf(log,"\nerr:no space for record!");
28             delay(3000);
29             exit(1);
30
31             fclose(log);
32             return 0;
33         }
34         fseek(frecord,i*sizeof(RECORD),SEEK_SET);
35         fread(rcd,sizeof(RECORD),1,frecord);
36
37         if(rcd->pakey==pakey&&rcd->trkey==trkey){
38             if (rcd!=NULL){
39                 free(rcd);
40                 rcd=NULL;
41             }
42
43             if (fclose(frecord)!=0){
44                 printf("\nerr:close record fail!");
45                 delay(3000);
46                 exit(1);
47                 return 0;
48             }
49
50             fprintf(log,"\ninfo:write record pakey %d trkey %d fail:already wr
itten!",pakey,trkey);
51             fclose(log);
52             return -1;
53         }
54
55         if (rcd!=NULL){
56             free(rcd);
57             rcd=NULL;
58         }
59     }
60
61     if((rcd=(RECORD*)malloc(sizeof(RECORD)))==NULL)
62     {
63         fprintf(log,"\nerr:no space for record!");
64         delay(3000);
65         exit(1);
66
67         fclose(log);
68         return 0;
69     }
70

```

```

70
71     rcd->pakey=pakey;
72     rcd->trkey=trkey;
73
74     fseek(frecord,0,SEEK_END);
75     fwrite(rcd,sizeof(RECORD),1,frecord);
76     fflush(frecord);
77
78     if (rcd != NULL)
79     {
80         free(rcd);
81         rcd = NULL;
82     }
83
84     if (fclose(frecord)!=0)
85     {
86         printf("\nerr:close record fail!");
87         delay(3000);
88         exit(1);
89         return 0;
90     }
91
92     fprintf(log,"\ninfo:write record pakey %d trkey %d success!",pakey,trkey);
93
94     fclose(log);
95     return 1;
96 }
97
98 int writerecordv1(int pakey,int trkey,char *track){
99     FILE* frecord;
100     FILE* log;
101     struct record* rcd;
102     int i;
103     int n;
104
105     log=fopen(".\\LOGGER","a+");
106
107     if((frecord=fopen(".\\DB\\RECORD", "rb+"))==NULL){
108         fprintf(log,"\nerr:open record fail! %s",strerror(errno));
109         delay(3000);
110         exit(1);
111
112         fclose(log);
113         return 0;
114     }
115
116     fseek(frecord,0,SEEK_END);
117     n=ftell(frecord)/sizeof(RECORD);

```

```

118
119     for(i=0;i<n;i++){
120         if ((rcd=(RECORD*)malloc(sizeof(RECORD)))==NULL){
121             fprintf(log,"\nerr:no space for record!");
122             delay(3000);
123             exit(1);
124
125             fclose(log);
126             return 0;
127         }
128         fseek(frecord,i*sizeof(RECORD),SEEK_SET);
129         fread(rcd,sizeof(RECORD),1,frecord);
130
131         if(rcd->pakey==pakey&&rcd->trkey==trkey){
132             if (rcd!=NULL){
133                 free(rcd);
134                 rcd=NULL;
135             }
136
137             if (fclose(frecord)!=0){
138                 printf("\nerr:close record fail!");
139                 delay(3000);
140                 exit(1);
141                 return 0;
142             }
143
144             fprintf(log,"\ninfo:write record pakey %d trkey %d fail:already wr
itten!",pakey,trkey);
145             fclose(log);
146             return -1;
147         }
148
149         if (rcd!=NULL){
150             free(rcd);
151             rcd=NULL;
152         }
153     }
154
155     if((rcd=(RECORD*)malloc(sizeof(RECORD)))==NULL)
156     {
157         fprintf(log,"\nerr:no space for record!");
158         delay(3000);
159         exit(1);
160
161         fclose(log);
162         return 0;
163     }
164

```

```

165     memset(rcd->track, '\\0', sizeof(rcd->track));
166     rcd->pakey=pakey;
167     rcd->trkey=trkey;
168     strcpy(rcd->track, track);
169
170     fseek(frecord, 0, SEEK_END);
171     fwrite(rcd, sizeof(RECORD), 1, frecord);
172     fflush(frecord);
173
174     if (rcd != NULL)
175     {
176         free(rcd);
177         rcd = NULL;
178     }
179
180     if (fclose(frecord)!=0)
181     {
182         printf("\\nerr:close record fail!");
183         delay(3000);
184         exit(1);
185         return 0;
186     }
187
188     fprintf(log, "\\ninfo:write record pakey %d trkey %d track %s success!", pake
y, trkey, track);
189
190     fclose(log);
191     return 1;
192 }
193
194 //>0 trkey个数 0 错误 -1 没有trkey
195 int getrecordbypakey(int pakey, int desttrkeyset[50]){
196     FILE* frecord;
197     FILE* log;
198     struct record* rcd;
199     int i;
200     int n;
201     int j=0;
202
203     log=fopen(".\\\\LOGGER", "a+");
204
205     if((frecord=fopen(".\\\\DB\\\\RECORD", "rb+" ))==NULL){
206         fprintf(log, "\\nerr:open record fail!");
207         delay(3000);
208         exit(1);
209
210         fclose(log);
211         return 0;

```

```

211         return 0;
212     }
213
214     fseek(frecord,0,SEEK_END);
215     n=ftell(frecord)/sizeof(RECORD);
216
217     for(i=0;i<n;i++){
218         if ((rcd=(RECORD*)malloc(sizeof(RECORD)))==NULL){
219             fprintf(log,"\nerr:no space for record!");
220             delay(3000);
221             exit(1);
222
223             fclose(log);
224             return 0;
225         }
226         fseek(frecord,i*sizeof(RECORD),SEEK_SET);
227         fread(rcd,sizeof(RECORD),1,frecord);
228
229         if(rcd->pakey==pakey){
230             desttrkeyset[j]=rcd->trkey;
231             j++;
232
233             if (rcd!=NULL){
234                 free(rcd);
235                 rcd=NULL;
236             }
237         }
238         if (rcd!=NULL){
239             free(rcd);
240             rcd=NULL;
241         }
242     }
243
244     if(j==0){
245         if (fclose(frecord)!= 0){
246             fprintf(log,"\nerr:close record fail!");
247             delay(3000);
248             exit(1);
249
250             fclose(log);
251             return 0;
252         }
253         fprintf(log,"\ninfo:get record trkey set by pakey %d fail:no pakey in
record!",pakey);
254
255         fclose(log);
256         return -1;
257     }

```



```

258
259     if (fclose(frecord) != 0){
260         fprintf(log, "\nerr:close record fail!");
261         delay(3000);
262         exit(1);
263
264         fclose(log);
265         return 0;
266     }
267
268     fprintf(log, "\ninfo:get record trkey set by pakey %d success:", pakey);
269     for(i=0; i<j; i++) fprintf(log, " %d", desttrkeyset[i]);
270     fprintf(log, "!");
271
272     fclose(log);
273     return j;
274 }
275
276 //>0 pakey个数 0 错误 -1 没有pakey
277 int getrecordbytrkey(int trkey, int destpakeyset[50]){
278     FILE* frecord;
279     FILE* log;
280     struct record* rcd;
281     int i;
282     int n;
283     int j=0;
284
285     log=fopen(".\\LOGGER", "a+");
286
287     if((frecord=fopen(".\\DB\\RECORD", "rb+"))==NULL){
288         fprintf(log, "\nerr:open record fail!");
289         delay(3000);
290         exit(1);
291
292         fclose(log);
293         return 0; //???
294     }
295
296     fseek(frecord, 0, SEEK_END);
297     n=ftell(frecord)/sizeof(RECORD);
298
299
300     for(i=0; i<n; i++){
301         if ((rcd=(RECORD*)malloc(sizeof(RECORD)))==NULL){
302             fprintf(log, "\nerr:no space for record!");
303             delay(3000);
304             exit(1);
305

```

```

306         fclose(log);
307         return 0;
308     }
309     fseek(frecord,i*sizeof(RECORD),SEEK_SET);
310     fread(rcd,sizeof(RECORD),1,frecord);
311
312     if(rcd->trkey==trkey){
313         destpakeset[j]=rcd->pakey;
314         j++;
315
316         if (rcd!=NULL){
317             free(rcd);
318             rcd=NULL;
319
320         }
321     }
322     if (rcd!=NULL){
323         free(rcd);
324         rcd=NULL;
325     }
326 }
327
328 if(j==0){
329     if (fclose(frecord)!= 0){
330         fprintf(log,"\nerr:close record fail!");
331         delay(3000);
332         exit(1);
333
334         fclose(log);
335         return 0;
336     }
337     fprintf(log,"\ninfo:get record pakey set by trkey %d fail:no pakey in
record!",trkey);
338
339     fclose(log);
340     return -1;
341 }
342
343 if (fclose(frecord)!= 0){
344     fprintf(log,"\nerr:close record fail!");
345     delay(3000);
346     exit(1);
347
348     fclose(log);
349     return 0;
350 }
351
352 if (fclose(log)!= 0){
353     fprintf(log,"\nerr:close log fail!");
354     delay(3000);
355     exit(1);
356 }

```

```

352     fprintf(log, "\ninfo: get record pakey set by trkey %d success:", trkey);
353     for(i=0; i<j; i++) fprintf(log, " %d", destpakeyset[i]);
354     fprintf(log, "!");
355
356     fclose(log);
357     return j;
358 }
359
360 int gettrackbypakey(int pakey, char trackset[10][10]){
361     FILE* frecord;
362     FILE* log;
363     struct record* rcd;
364     int trackcount=0;
365     int i;
366     int n;
367
368     log=fopen(".\\LOGGER", "a+");
369
370     if((frecored=fopen(".\\DB\\RECORD", "rb+" ))==NULL){
371         fprintf(log, "\nerr: open record fail!");
372         delay(3000);
373         exit(1);
374
375         fclose(log);
376         return 0; //???
377     }
378
379     fseek(frecord, 0, SEEK_END);
380     n=ftell(frecord)/sizeof(RECORD);
381
382     for(i=0; i<n; i++){
383         if ((rcd=(RECORD*)malloc(sizeof(RECORD)))==NULL){
384             fprintf(log, "\nerr: no space for record!");
385             delay(3000);
386             exit(1);
387
388             fclose(log);
389             return 0;
390         }
391         fseek(frecord, i*sizeof(RECORD), SEEK_SET);
392         fread(rcd, sizeof(RECORD), 1, frecored);
393
394         if(rcd->pakey==pakey){
395             strcpy(trackset[trackcount], rcd->track);
396             trackcount++;
397
398             if (rcd!=NULL){
399                 free(rcd);

```

```

400         rcd=NULL;
401     }
402 }
403 if (rcd!=NULL){
404     free(rcd);
405     rcd=NULL;
406 }
407 }
408
409 if(trackcount==0){
410     if (fclose(frecord)!= 0){
411         fprintf(log,"\nerr:close record fail!");
412         delay(3000);
413         exit(1);
414
415         fclose(log);
416         return 0;
417     }
418     fprintf(log,"\ninfo:get record track set by pakey %d fail:no track in
record!",pakey);
419
420     fclose(log);
421     return -1;
422 }
423
424 if (fclose(frecord)!= 0){
425     fprintf(log,"\nerr:close record fail!");
426     delay(3000);
427     exit(1);
428
429     fclose(log);
430     return 0;
431 }
432
433 fprintf(log,"\ninfo:get track set by pakey %d success:",pakey);
434 for(i=0;i<trackcount;i++)fprintf(log," %s",trackset[i]);
435 fprintf(log,"!");
436
437 fclose(log);
438 return trackcount;
439 }

```

7. filetxt.c

C

```
1 #include<stdio.h>
```

```

1  #include "public.h"
2
3  int getcnamebyID(char *ID, char *cname){
4      FILE *fcnametxt;
5      FILE *log;
6      char tempID[20];
7
8      log=fopen(".\\LOGGER", "a+");
9
10     if((fcnametxt=fopen(".\\DB\\cname.txt", "r+" ))==NULL){
11         fprintf(log, "\\nerr:open cname.txt fail!");
12         delay(3000);
13         exit(1);
14
15         fclose(log);
16         return 0;
17     }
18
19     while(!feof(fcnametxt)){
20         memset(tempID, '\\0', sizeof(ID));
21         memset(cname, '\\0', sizeof(cname));
22         fscanf(fcnametxt, "%s", tempID);
23         fgetc(fcnametxt);
24         fscanf(fcnametxt, "%s", cname);
25         fgetc(fcnametxt);
26         if(strcmp(tempID, ID)==0){
27             if (fclose(fcnametxt)!=0){
28                 fprintf(log, "\\nerr:close cname.txt fail!");
29                 delay(3000);
30                 exit(1);
31
32                 fclose(log);
33                 return 0;
34             }
35
36             fclose(log);
37             return 1;
38         }
39     }
40
41     if (fclose(fcnametxt)!=0){
42         fprintf(log, "\\nerr:close cname.txt fail!");
43         delay(3000);
44         exit(1);
45
46         fclose(log);
47         return 0;
48     }

```

```
49
50     fclose(log);
51     return 0;
52 }
53
54 int getstartbytrainname(char *trainname, char *start){
55     FILE *fstarttxt;
56     FILE *log;
57     char temptrainname[10];
58
59     log=fopen(".\\LOGGER", "a+");
60
61     if((fstarttxt=fopen(".\\DB\\start.txt", "r+" ))==NULL){
62         fprintf(log, "\\nerr:open start.txt fail!");
63         delay(3000);
64         exit(1);
65
66         fclose(log);
67         return 0;
68     }
69
70     while(!feof(fstarttxt)){
71         memset(start, '\\0', sizeof(start));
72         memset(temptrainname, '\\0', sizeof(temptrainname));
73         fscanf(fstarttxt, "%s", temptrainname);
74         fgetc(fstarttxt);
75         fscanf(fstarttxt, "%s", start);
76         fgetc(fstarttxt);
77         if(strcmp(temptrainname, trainname)==0){
78             if (fclose(fstarttxt)!=0){
79                 fprintf(log, "\\nerr:close start.txt fail!");
80                 delay(3000);
81                 exit(1);
82
83                 fclose(log);
84                 return 0;
85             }
86
87             fclose(log);
88             return 1;
89         }
90     }
91
92     if (fclose(fstarttxt)!=0){
93         fprintf(log, "\\nerr:close start.txt fail!");
94         delay(3000);
95         exit(1);
96     }
```

```

97         fclose(log);
98         return 0;
99     }
100
101     fclose(log);
102     return -1;
103 }
104
105 int getendbytrainname(char *trainname, char *end){
106     FILE *fendtxt;
107     FILE *log;
108     char temptrainname[10];
109
110     log=fopen(".\\LOGGER", "a+");
111
112     if((fendtxt=fopen(".\\DB\\end.txt", "r+ "))==NULL){
113         fprintf(log, "\\nerr:open end.txt fail!");
114         delay(3000);
115         exit(1);
116
117         fclose(log);
118         return 0;
119     }
120
121     while(!feof(fendtxt)){
122         memset(end, '\\0', sizeof(end));
123         memset(temptrainname, '\\0', sizeof(temptrainname));
124         fscanf(fendtxt, "%s", temptrainname);
125         fgetc(fendtxt);
126         fscanf(fendtxt, "%s", end);
127         fgetc(fendtxt);
128         if(strcmp(temptrainname, trainname)==0){
129             if (fclose(fendtxt)!=0){
130                 fprintf(log, "\\nerr:close end.txt fail!");
131                 delay(3000);
132                 exit(1);
133
134                 fclose(log);
135                 return 0;
136             }
137
138             fclose(log);
139             return 1;
140         }
141     }
142
143     if (fclose(fendtxt)!=0){

```

```

144         fprintf(log, "\\nerr:close end.txt fail!");
145         delay(3000);
146         exit(1);
147
148         fclose(log);
149         return 0;
150     }
151
152     fclose(log);
153     return -1;
154 }

```

8. graphpro.c

```

C

1  #include <graphics.h>
2  #include "hz.h"
3  #include "public.h"
4  #include <stdio.h>
5  #include <stdlib.h>
6
7  void drawtop(void){
8      setfillstyle(1,BLUE);
9      bar(0,0,640,40);
10     puthz(160,4,"公共交通行程管理系统",32,32,LIGHTGRAY);
11
12     setcolor(RED);
13     setfillstyle(1,RED);
14     pieslice(560,18,315,360,13);
15     pieslice(560,18,0,225,13);
16     bar(557,3,563,8);
17     setcolor(BLUE);
18     setfillstyle(1,BLUE);
19     pieslice(560,18,315,360,10);
20     pieslice(560,18,0,225,10);
21     setcolor(RED);
22     setfillstyle(1,RED);
23     bar(553,16,567,20);
24     bar(559,20,561,30);
25     bar(548,30,572,32);
26
27     return;
28 }
29
30 void barword16(int left,int top,int barcolor,int width,int n,char *s,int wordc
olor){

```



```

31     int right=width*16+4+left;
32     int bottom=20+top;
33     float m=n/2.0;
34     float middle=(left+right)/2.0;
35
36     if(left==0){
37         setfillstyle(1,barcolor);
38         bar(320-8*width-2,top,320+8*width+2,bottom);
39         puthz(320-16*m,top+2,s,16,16,wordcolor);
40         return;
41     }//位置居中
42
43     setfillstyle(1,barcolor);
44     bar(left,top,right,bottom);
45     puthz(middle-16*m,top+2,s,16,16,wordcolor);
46     return;
47 }
48 void barword24(int left,int top,int barcolor,int width,int n,char *s,int wordc
olor){
49     int right=width*24+6+left;
50     int bottom=30+top;
51     float m=n/2.0;
52     float middle=(left+right)/2.0;
53
54     if(left==0){
55         setfillstyle(1,barcolor);
56         bar(320-12*width-3,top,320+12*width+3,bottom);
57         puthz(320-12*m,top+3,s,24,24,wordcolor);
58         return;
59     }//位置居中
60
61     setfillstyle(1,barcolor);
62     bar(left,top,right,bottom);
63     puthz(middle-24*m,top+3,s,24,24,wordcolor);
64     return;
65 }
66 void barword32(int left,int top,int barcolor,int width,int n,char *s,int wordc
olor){
67     int right=width*32+8+left;
68     int bottom=40+top;
69     float m=n/2.0;
70     float middle=(left+right)/2.0;
71
72     if(left==0){
73         setfillstyle(1,barcolor);
74         bar(320-16*width-4,top,320+16*width+4,bottom);
75         puthz(320-32*m,top+4,s,32,32,wordcolor);
76

```

```

76         return;
77     }//位置居中
78
79     setfillstyle(1,barcolor);
80     bar(left,top,right,bottom);
81     puthz(middle-32*m,top+4,s,32,32,wordcolor);
82     return;
83 }
84 void barword48(int left,int top,int barcolor,int width,int n,char *s,int wordcolor){
85     int right=width*48+6+left;
86     int bottom=60+top;
87     float m=n/2.0;
88     float middle=(left+right)/2.0;
89
90     if(left==0){
91         setfillstyle(1,barcolor);
92         bar(320-24*width-6,top,320+24*width+6,bottom);
93         puthz(320-48*m,top+6,s,48,48,wordcolor);
94         return;
95     }//位置居中
96
97     setfillstyle(1,barcolor);
98     bar(left,top,right,bottom);
99     puthz(middle-48*m,top+6,s,48,48,wordcolor);
100    return;
101 }
102
103 void barwordframe(int left,int top,int barcolor,int width,char *s,int wordcolor,int framecolor){
104     int right=width*32+8+left;
105     int bottom=40+top;
106     float m=width/2.0;
107     int framewidth=1;
108
109     setfillstyle(1,framecolor);
110     bar(left,top,right,bottom);
111
112     setfillstyle(1,barcolor);
113     bar(left+framewidth,top+framewidth,right-framewidth,bottom-framewidth);
114
115     puthz(left+4,top+4,s,32,32,wordcolor);
116     return;
117 }
118
119 void returnbutton(int color){//返回按钮
120     int x1=10,x2=30,x3=30,y1=20,y2=10,y3=30;
121     setcolor(color);

```

```
122
123     setlinestyle(SOLID_LINE,0,3);
124     line(x1,y1,x2,y2);
125     line(x1,y1,x3,y3);
126
127     return;
128 }
129 void searchbutton(int color){
130     setcolor(color);
131     setfillstyle(1,BLUE);
132     setlinestyle(0,0,3);
133     bar(510,100,550,140);
134     circle(535,115,10);
135     line(515,132,527,120);
136 }
137
138 void loginok(void){
139     barword32(0,200,WHITE,5,4,"登录成功",GREEN);
140     return;
141 }
142 void loginfail(void){
143     barword32(0,200,WHITE,5,4,"登录失败",RED);
144     return;
145 }
146 void loginpasswordwrong(void){
147     barword32(0,200,WHITE,5,4,"密码错误",RED);
148 }
149 void loginnoregister(void){
150     barword32(0,200,WHITE,7,6,"该账号未注册",RED);
151 }
152
153 void registerok(void){
154     barword32(0,225,BLUE,4,4,"注册成功",GREEN);
155     return;
156 }
157 void registerrepeat(void){
158     barword32(0,225,BLUE,5,5,"账号已注册",RED);
159     return;
160 }
161 void registerpasswrong(void){
162     barword32(0,225,BLUE,6,6,"两次密码不同",RED);
163     return;
164 }
165 void registerpasszero(void){
166     barword32(0,225,BLUE,6,6,"密码不得为空",RED);
167     return;
168 }
169 void registernamezero(void){
```

```
170     barword32(0,225,BLUE,6,6,"账号不得为空",RED);
171     return;
172 }
173 void registerpassshort(void){
174     barword32(0,225,BLUE,10,10,"密码长度不得少于六位",RED);
175     return;
176 }
177
178 void bindok(void){
179     barword32(0,225,BLUE,4,4,"绑定成功",GREEN);
180     return;
181 }
182 void bindIDrepeat(void){
183     barword32(0,225,BLUE,7,7,"此身份证已绑定",RED);
184     return;
185 }
186 void bindIDwrong(void){
187     barword32(0,225,BLUE,7,7,"身份证号不合法",RED);
188     return;
189 }
190 void bindtelwrong(void){
191     barword32(0,225,BLUE,7,7,"电话号码不合法",RED);
192     return;
193 }
194
195 void labelok(void){
196     barword32(0,225,BLUE,4,4,"录入成功",GREEN);
197     return;
198 }
199 void labelIDzero(void){
200     barword32(0,225,BLUE,8,8,"后台无该身份证号",RED);
201     return;
202 }
203
204 void statusabnormal(void){
205     barword32(0,225,BLUE,6,6,"健康状态异常",RED);
206     return;
207 }
208
209 void recordzero(void){
210     barword32(0,225,BLUE,4,4,"无该车次",RED);
211     return;
212 }
213
214 void trackrecordrepeat(void){
215     barword32(0,225,BLUE,4,4,"重复录入",RED);
216     return;
```

```

217 }
218
219 void trackrecordok(void){
220     barword32(0,225,BLUE,4,4,"录入成功",RED);
221     return;
222 }
223
224 void palabelu(struct passenger *pa){
225     int x1=60;
226     int y1=130;
227     int x2=x1+520;
228     int y2=80+y1;
229     int x;
230     char cname[5];
231     getcnamebyID(pa->ID,cname);
232
233     setfillstyle(1,BLUE);
234     bar(x1,y1,x2,y2);
235     puthz(x1+20,y1+10,"姓名",16,16,LIGHTGRAY);
236     puthz(x1+20,y1+50,"状态",16,16,LIGHTGRAY);
237     puthz(x1+180,y1+10,"电话",16,16,LIGHTGRAY);
238     puthz(x1+180,y1+50,"身份证号",16,16,LIGHTGRAY);
239
240     puthz(x1+80,y1+10,cname,16,16,LIGHTGRAY);
241     setcolor(LIGHTGRAY);
242     settextstyle(1,0,2);
243     outtextxy(x1+260,y1+6,pa->tel);
244     outtextxy(x1+260,y1+44,pa->ID);
245
246     x=pa->status;
247     if(x==0)puthz(x1+80,y1+50,"健康",16,16,GREEN);
248     else if(x==1)puthz(x1+80,y1+50,"新冠肺炎患者",16,16,RED);
249     else if(x==2)puthz(x1+80,y1+50,"无症状感染者",16,16,RED);
250     else if(x==3)puthz(x1+80,y1+50,"密切接触者",16,16,YELLOW);
251 }
252 void palabelm(struct passenger *pa){
253     int x1=60;
254     int y1=230;
255     int x2=x1+520;
256     int y2=80+y1;
257     int x;
258     char cname[5];
259     getcnamebyID(pa->ID,cname);
260
261     setfillstyle(1,BLUE);
262     bar(x1,y1,x2,y2);
263     puthz(x1+20,y1+10,"姓名",16,16,LIGHTGRAY);
264     puthz(x1+20,y1+50,"状态",16,16,LIGHTGRAY);

```

```

265     puthz(x1+180,y1+10,"电话",16,16,LIGHTGRAY);
266     puthz(x1+180,y1+50,"身份证号",16,16,LIGHTGRAY);
267
268     puthz(x1+80,y1+10,cname,16,16,LIGHTGRAY);
269     setcolor(LIGHTGRAY);
270     settextstyle(1,0,2);
271     outtextxy(x1+260,y1+6,pa->tel);
272     outtextxy(x1+260,y1+44,pa->ID);
273     x=pa->status;
274
275     if(x==0)puthz(x1+80,y1+50,"健康",16,16,GREEN);
276     else if(x==1)puthz(x1+80,y1+50,"新冠肺炎患者",16,16,RED);
277     else if(x==2)puthz(x1+80,y1+50,"无症状感染者",16,16,RED);
278     else if(x==3)puthz(x1+80,y1+50,"密切接触者",16,16,YELLOW);
279 }
280 void palabell(struct passenger *pa){
281     int x1=60;
282     int y1=330;
283     int x2=x1+520;
284     int y2=80+y1;
285     int x;
286     char cname[5];
287     getcnamebyID(pa->ID,cname);
288
289     setfillstyle(1,BLUE);
290     bar(x1,y1,x2,y2);
291     puthz(x1+20,y1+10,"姓名",16,16,LIGHTGRAY);
292     puthz(x1+20,y1+50,"状态",16,16,LIGHTGRAY);
293     puthz(x1+180,y1+10,"电话",16,16,LIGHTGRAY);
294     puthz(x1+180,y1+50,"身份证号",16,16,LIGHTGRAY);
295
296     puthz(x1+80,y1+10,cname,16,16,LIGHTGRAY);
297     setcolor(LIGHTGRAY);
298     settextstyle(1,0,2);
299     outtextxy(x1+260,y1+6,pa->tel);
300     outtextxy(x1+260,y1+44,pa->ID);
301
302     x=pa->status;
303     if(x==0)puthz(x1+80,y1+50,"健康",16,16,GREEN);
304     else if(x==1)puthz(x1+80,y1+50,"新冠肺炎患者",16,16,RED);
305     else if(x==2)puthz(x1+80,y1+50,"无症状感染者",16,16,RED);
306     else if(x==3)puthz(x1+80,y1+50,"密切接触者",16,16,YELLOW);
307 }
308
309 void trlabelu(struct pagetrain *trn){
310     int x1=60;
311     int y1=130;
312     int x2=x1+520;

```

```

312     int x2=x1+520;
313     int y2=80+y1;
314     int yy,mm,dd;
315     char start[5];
316     char end[5];
317     getstartbytrainname(trn->trainname,start);
318     getendbytrainname(trn->trainname,end);
319
320     setfillstyle(1,BLUE);
321     bar(x1,y1,x2,y2);
322     puthz(x1+20,y1+10,"车次",16,16,LIGHTGRAY);
323     puthz(x1+20,y1+50,"日期",16,16,LIGHTGRAY);
324     puthz(x1+180,y1+10,"始发",16,16,LIGHTGRAY);
325     puthz(x1+330,y1+10,"终点",16,16,LIGHTGRAY);
326     puthz(x1+330,y1+50,"状态",16,16,LIGHTGRAY);
327
328     getdate(trn->date,&yy,&mm,&dd);
329     setcolor(LIGHTGRAY);
330     settextstyle(1,0,2);
331     outtextxy(x1+100,y1+6,trn->trainname);
332     puthz(x1+220,y1+10,start,16,16,LIGHTGRAY);
333     puthz(x1+370,y1+10,end,16,16,LIGHTGRAY);
334     outtextxy(x1+100,y1+46,datestring(trn->date));
335     if(trn->status==0){
336         puthz(x1+370,y1+50,"正常",16,16,GREEN);
337     }
338     else if(trn->status==1){
339         puthz(x1+370,y1+50,"异常",16,16,RED);
340     }
341 }
342 void trlabelm(struct pagetrain *trn){
343     int x1=60;
344     int y1=230;
345     int x2=x1+520;
346     int y2=80+y1;
347     int yy,mm,dd;
348     char start[5];
349     char end[5];
350     getstartbytrainname(trn->trainname,start);
351     getendbytrainname(trn->trainname,end);
352
353     setfillstyle(1,BLUE);
354     bar(x1,y1,x2,y2);
355     puthz(x1+20,y1+10,"车次",16,16,LIGHTGRAY);
356     puthz(x1+20,y1+50,"日期",16,16,LIGHTGRAY);
357     puthz(x1+180,y1+10,"始发",16,16,LIGHTGRAY);
358     puthz(x1+330,y1+10,"终点",16,16,LIGHTGRAY);
359     puthz(x1+330,y1+50,"状态",16,16,LIGHTGRAY);

```

```

360
361     getdate(trn->date,&yy,&mm,&dd);
362     setcolor(LIGHTGRAY);
363     settextstyle(1,0,2);
364     outtextxy(x1+100,y1+6,trn->trainname);
365     puthz(x1+220,y1+10,start,16,16,LIGHTGRAY);
366     puthz(x1+370,y1+10,end,16,16,LIGHTGRAY);
367     outtextxy(x1+100,y1+46,datestring(trn->date));
368     if(trn->status==0){
369         puthz(x1+370,y1+50,"正常",16,16,GREEN);
370     }
371     else if(trn->status==1){
372         puthz(x1+370,y1+50,"异常",16,16,RED);
373     }
374 }
375 void trlabel1(struct pagetrain *trn){
376     int x1=60;
377     int y1=330;
378     int x2=x1+520;
379     int y2=80+y1;
380     int yy,mm,dd;
381     char start[5];
382     char end[5];
383     getstartbytrainname(trn->trainname,start);
384     getendbytrainname(trn->trainname,end);
385
386     setfillstyle(1,BLUE);
387     bar(x1,y1,x2,y2);
388     puthz(x1+20,y1+10,"车次",16,16,LIGHTGRAY);
389     puthz(x1+20,y1+50,"日期",16,16,LIGHTGRAY);
390     puthz(x1+180,y1+10,"始发",16,16,LIGHTGRAY);
391     puthz(x1+330,y1+10,"终点",16,16,LIGHTGRAY);
392     puthz(x1+330,y1+50,"状态",16,16,LIGHTGRAY);
393
394     getdate(trn->date,&yy,&mm,&dd);
395     setcolor(LIGHTGRAY);
396     settextstyle(1,0,2);
397     outtextxy(x1+100,y1+6,trn->trainname);
398     puthz(x1+220,y1+10,start,16,16,LIGHTGRAY);
399     puthz(x1+370,y1+10,end,16,16,LIGHTGRAY);
400     outtextxy(x1+100,y1+46,datestring(trn->date));
401     if(trn->status==0){
402         puthz(x1+370,y1+50,"正常",16,16,GREEN);
403     }
404     else if(trn->status==1){
405         puthz(x1+370,y1+50,"异常",16,16,RED);
406     }
407 }

```



```

408
409 void leftarrow(int color){
410     setcolor(color);
411     line(270,450,270,430);
412     line(270,430,250,440);
413     line(270,450,250,440);
414 }
415 void rightarrow(int color){
416     setcolor(color);
417     line(370,430,370,450);
418     line(370,430,390,440);
419     line(370,450,390,440);
420 }
421
422 void pagenumber(int currentpage,int countpage){
423     char currentstr[3];
424     char countstr[3];
425
426     currentstr[0]=currentpage+'0';
427     currentstr[1]='\0';
428     countstr[0]=countpage+'0';
429     countstr[1]='\0';
430
431     settextstyle(1,0,3);
432     setcolor(BLUE);
433     outtextxy(320-22,425,currentstr);
434     outtextxy(320+14,425,countstr);
435     setlinestyle(SOLID_LINE,0,3);
436     line(325,430,320-5,450);
437 }
438
439 void putnum(int x,int y,int num){
440     char *numstring;
441     itoa(num,numstring,10);
442     outtextxy(x,y,numstring);
443     free(numstring);
444     return;
445 }
446
447 void todaydate(void){
448     int yy;
449     int mm;
450     int dd;
451     char Y[5]={'\0'};
452     char M[5]={'\0'};
453     char D[5]={'\0'};
454     puthz(350,370,"乘车日期为",16,16,RED);
455     puthz(500,370,"年",16,16,RED);

```

```

455     puthz(500,370,"+",16,16,RED);
456     puthz(543,370,"月",16,16,RED);
457     puthz(586,370,"日",16,16,RED);
458     gettodaydate(&yy,&mm,&dd);
459     setcolor(RED);
460     settextstyle(1, 0, 1);
461
462     itoa(yy,Y,10);
463     itoa(mm,M,10);
464     itoa(dd,D,10);
465     outtextxy(443,365,Y);
466     outtextxy(525,365,M);
467     outtextxy(560,365,D);
468
469     return;
470 }
471
472 void position1(int x,int y,int *s){           //名字在上角
473     setcolor(GREEN);
474     setfillstyle(1,GREEN);
475     pieslice(x,y,0,360,3);
476     setcolor(GREEN);
477     line(x,y,x+3,y);
478     puthz(x,y-25,s,16,16,RED);
479 }
480
481 void position2(int x,int y,int *s){           //名字在右下角
482     setcolor(GREEN);
483     setfillstyle(1,GREEN);
484     pieslice(x,y,0,360,3);
485     setcolor(GREEN);
486     line(x,y,x+3,y);
487     puthz(x,y+15,s,16,16,RED);
488 }
489
490 void drawmap(void){
491     position1(425,120,"北京");
492     position1(325,165,"太原");
493     position1(155,210,"西安");
494     position1(55,270,"成都");
495     position1(95,310,"重庆");
496     position1(305,270,"武汉");
497     position1(465,240,"南京");
498     position2(485,280,"杭州");
499     position1(505,250,"苏州");
500     position2(565,252,"上海");
501     position1(209,420,"广州");
502     position1(229,440,"深圳");

```

```

503     position1(385,330,"南昌");
504     position1(200,350,"长沙");
505 }
506
507 void mapline(char *s,int color){
508     setlinestyle(0,1,3);
509     setcolor(color);
510     do {
511         line(xpos(*s),ypos(*s),xpos(*(s+1)),ypos(*(s+1)));
512         s++;
513     }while(*(s+1)!='\0');
514     return;
515 }
516
517 void track1(char *trainname,int color){
518     if(strcmp(trainname,"G562")==0)barword24(110,190,BLUE,6,6,"北京》》太原",color);
519     if(strcmp(trainname,"G567")==0)barword24(110,190,BLUE,6,6,"上海》》杭州",color);
520     if(strcmp(trainname,"G751")==0)barword24(110,190,BLUE,6,6,"南昌》》武汉",color);
521     if(strcmp(trainname,"G768")==0)barword24(110,190,BLUE,6,6,"广州》》重庆",color);
522     if(strcmp(trainname,"G267")==0)barword24(110,190,BLUE,6,6,"深圳》》广州",color);
523     if(strcmp(trainname,"G186")==0)barword24(110,190,BLUE,6,6,"南京》》武汉",color);
524     if(strcmp(trainname,"G379")==0)barword24(110,190,BLUE,6,6,"杭州》》南昌",color);
525     if(strcmp(trainname,"G467")==0)barword24(110,190,BLUE,6,6,"西安》》武汉",color);
526     if(strcmp(trainname,"G685")==0)barword24(110,190,BLUE,6,6,"苏州》》南昌",color);
527     if(strcmp(trainname,"G335")==0)barword24(110,190,BLUE,6,6,"北京》》武汉",color);
528 }
529 void track2(char *trainname,int color){
530     if(strcmp(trainname,"G562")==0)barword24(110,246,BLUE,6,6,"北京》》南京",color);
531     if(strcmp(trainname,"G567")==0)barword24(110,246,BLUE,6,6,"上海》》南昌",color);
532     if(strcmp(trainname,"G751")==0)barword24(110,246,BLUE,6,6,"南昌》》南京",color);
533     if(strcmp(trainname,"G768")==0)barword24(110,246,BLUE,6,6,"广州》》成都",color);
534     if(strcmp(trainname,"G267")==0)barword24(110,246,BLUE,6,6,"深圳》》长沙",color);
535     if(strcmp(trainname,"G186")==0)barword24(110,246,BLUE,6,6,"南京》》长沙",color);

```

```

or);
536     if(strcmp(trainname,"G379")==0)barword24(110,246,BLUE,6,6,"杭州》》武汉",col
or);
537     if(strcmp(trainname,"G467")==0)barword24(110,246,BLUE,6,6,"西安》》南昌",col
or);
538     if(strcmp(trainname,"G685")==0)barword24(110,246,BLUE,6,6,"苏州》》北京",col
or);
539     if(strcmp(trainname,"G335")==0)barword24(110,246,BLUE,6,6,"北京》》长沙",col
or);
540 }
541 void track3(char *trainname,int color){
542     if(strcmp(trainname,"G562")==0)barword24(110,302,BLUE,6,6,"北京》》苏州",col
or);
543     if(strcmp(trainname,"G567")==0)barword24(110,302,BLUE,6,6,"上海》》深圳",col
or);
544     if(strcmp(trainname,"G751")==0)barword24(110,302,BLUE,6,6,"南昌》》太原",col
or);
545     if(strcmp(trainname,"G768")==0)barword24(110,302,BLUE,6,6,"广州》》西安",col
or);
546     if(strcmp(trainname,"G267")==0)barword24(110,302,BLUE,6,6,"深圳》》武汉",col
or);
547     if(strcmp(trainname,"G186")==0)barword24(110,302,BLUE,6,6,"南京》》重庆",col
or);
548     if(strcmp(trainname,"G379")==0)barword24(110,302,BLUE,6,6,"杭州》》西安",col
or);
549     if(strcmp(trainname,"G467")==0)barword24(110,302,BLUE,6,6,"西安》》杭州",col
or);
550     if(strcmp(trainname,"G685")==0)barword24(110,302,BLUE,6,6,"苏州》》太原",col
or);
551     if(strcmp(trainname,"G335")==0)barword24(110,302,BLUE,6,6,"北京》》广州",col
or);
552 }
553 void track4(char *trainname,int color){
554     if(strcmp(trainname,"G562")==0)barword24(110,358,BLUE,6,6,"北京》》上海",col
or);
555     if(strcmp(trainname,"G567")==0)barword24(110,358,BLUE,6,6,"上海》》广州",col
or);
556     if(strcmp(trainname,"G751")==0)barword24(110,358,BLUE,6,6,"南昌》》北京",col
or);
557     if(strcmp(trainname,"G768")==0)barword24(110,358,BLUE,6,6,"广州》》太原",col
or);
558     if(strcmp(trainname,"G267")==0)barword24(110,358,BLUE,6,6,"深圳》》北京",col
or);
559     if(strcmp(trainname,"G186")==0)barword24(110,358,BLUE,6,6,"南京》》成都",col
or);
560     if(strcmp(trainname,"G379")==0)barword24(110,358,BLUE,6,6,"杭州》》成都",col
or);
561     if(strcmp(trainname,"G467")==0)barword24(110,358,BLUE,6,6,"西安》》上海",col
or);

```

```
561     if(strcmp(trainname,"G467")==0)barword24(110,358,BLUE,6,6,"四女》》上海",color);
562     if(strcmp(trainname,"G685")==0)barword24(110,358,BLUE,6,6,"苏州》》西安",color);
563     if(strcmp(trainname,"G335")==0)barword24(110,358,BLUE,6,6,"北京》》深圳",color);
564 }
565 void track5(char *trainname,int color){
566     if(strcmp(trainname,"G562")==0)barword24(110,414,BLUE,6,6,"太原》》南京",color);
567     if(strcmp(trainname,"G567")==0)barword24(110,414,BLUE,6,6,"杭州》》南昌",color);
568     if(strcmp(trainname,"G751")==0)barword24(110,414,BLUE,6,6,"武汉》》南京",color);
569     if(strcmp(trainname,"G768")==0)barword24(110,414,BLUE,6,6,"重庆》》成都",color);
570     if(strcmp(trainname,"G267")==0)barword24(110,414,BLUE,6,6,"广州》》长沙",color);
571     if(strcmp(trainname,"G186")==0)barword24(110,414,BLUE,6,6,"武汉》》长沙",color);
572     if(strcmp(trainname,"G379")==0)barword24(110,414,BLUE,6,6,"南昌》》武汉",color);
573     if(strcmp(trainname,"G467")==0)barword24(110,414,BLUE,6,6,"武汉》》南昌",color);
574     if(strcmp(trainname,"G685")==0)barword24(110,414,BLUE,6,6,"南昌》》北京",color);
575     if(strcmp(trainname,"G335")==0)barword24(110,414,BLUE,6,6,"武汉》》长沙",color);
576 }
577 void track6(char *trainname,int color){
578     if(strcmp(trainname,"G562")==0)barword24(380,190,BLUE,6,6,"太原》》苏州",color);
579     if(strcmp(trainname,"G567")==0)barword24(380,190,BLUE,6,6,"杭州》》深圳",color);
580     if(strcmp(trainname,"G751")==0)barword24(380,190,BLUE,6,6,"武汉》》太原",color);
581     if(strcmp(trainname,"G768")==0)barword24(380,190,BLUE,6,6,"重庆》》西安",color);
582     if(strcmp(trainname,"G267")==0)barword24(380,190,BLUE,6,6,"广州》》武汉",color);
583     if(strcmp(trainname,"G186")==0)barword24(380,190,BLUE,6,6,"武汉》》重庆",color);
584     if(strcmp(trainname,"G379")==0)barword24(380,190,BLUE,6,6,"南昌》》西安",color);
585     if(strcmp(trainname,"G467")==0)barword24(380,190,BLUE,6,6,"武汉》》杭州",color);
586     if(strcmp(trainname,"G685")==0)barword24(380,190,BLUE,6,6,"南昌》》太原",color);
```

```
587     if(strcmp(trainname,"G335")==0)barword24(380,190,BLUE,6,6,"武汉》》广州",col
    or);
588 }
589 void track7(char *trainname,int color){
590     if(strcmp(trainname,"G562")==0)barword24(380,246,BLUE,6,6,"太原》》上海",col
    or);
591     if(strcmp(trainname,"G567")==0)barword24(380,246,BLUE,6,6,"杭州》》广西",col
    or);
592     if(strcmp(trainname,"G751")==0)barword24(380,246,BLUE,6,6,"武汉》》北京",col
    or);
593     if(strcmp(trainname,"G768")==0)barword24(380,246,BLUE,6,6,"重庆》》太原",col
    or);
594     if(strcmp(trainname,"G267")==0)barword24(380,246,BLUE,6,6,"广州》》北京",col
    or);
595     if(strcmp(trainname,"G186")==0)barword24(380,246,BLUE,6,6,"武汉》》成都",col
    or);
596     if(strcmp(trainname,"G379")==0)barword24(380,246,BLUE,6,6,"南昌》》成都",col
    or);
597     if(strcmp(trainname,"G467")==0)barword24(380,246,BLUE,6,6,"武汉》》上海",col
    or);
598     if(strcmp(trainname,"G685")==0)barword24(380,246,BLUE,6,6,"南昌》》西安",col
    or);
599     if(strcmp(trainname,"G335")==0)barword24(380,246,BLUE,6,6,"武汉》》深圳",col
    or);
600
601 }
602 void track8(char *trainname,int color){
603     if(strcmp(trainname,"G562")==0)barword24(380,302,BLUE,6,6,"南京》》苏州",col
    or);
604     if(strcmp(trainname,"G567")==0)barword24(380,302,BLUE,6,6,"南昌》》深圳",col
    or);
605     if(strcmp(trainname,"G751")==0)barword24(380,302,BLUE,6,6,"南京》》太原",col
    or);
606     if(strcmp(trainname,"G768")==0)barword24(380,302,BLUE,6,6,"成都》》西安",col
    or);
607     if(strcmp(trainname,"G267")==0)barword24(380,302,BLUE,6,6,"长沙》》武汉",col
    or);
608     if(strcmp(trainname,"G186")==0)barword24(380,302,BLUE,6,6,"长沙》》重庆",col
    or);
609     if(strcmp(trainname,"G379")==0)barword24(380,302,BLUE,6,6,"武汉》》西安",col
    or);
610     if(strcmp(trainname,"G467")==0)barword24(380,302,BLUE,6,6,"南昌》》杭州",col
    or);
611     if(strcmp(trainname,"G685")==0)barword24(380,302,BLUE,6,6,"北京》》太原",col
    or);
612     if(strcmp(trainname,"G335")==0)barword24(380,302,BLUE,6,6,"长沙》》广州",col
    or);
613
```

```
614 }
615 void track9(char *trainname,int color){
616     if(strcmp(trainname,"G562")==0)barword24(380,358,BLUE,6,6,"南京》》上海",color);
617     if(strcmp(trainname,"G567")==0)barword24(380,358,BLUE,6,6,"南昌》》广州",color);
618     if(strcmp(trainname,"G751")==0)barword24(380,358,BLUE,6,6,"南京》》北京",color);
619     if(strcmp(trainname,"G768")==0)barword24(380,358,BLUE,6,6,"成都》》太原",color);
620     if(strcmp(trainname,"G267")==0)barword24(380,358,BLUE,6,6,"长沙》》北京",color);
621     if(strcmp(trainname,"G186")==0)barword24(380,358,BLUE,6,6,"长沙》》成都",color);
622     if(strcmp(trainname,"G379")==0)barword24(380,358,BLUE,6,6,"武汉》》成都",color);
623     if(strcmp(trainname,"G467")==0)barword24(380,358,BLUE,6,6,"南昌》》上海",color);
624     if(strcmp(trainname,"G685")==0)barword24(380,358,BLUE,6,6,"北京》》西安",color);
625     if(strcmp(trainname,"G335")==0)barword24(380,358,BLUE,6,6,"长沙》》深圳",color);
626 }
627 void track10(char *trainname,int color){
628     if(strcmp(trainname,"G562")==0)barword24(380,414,BLUE,6,6,"苏州》》上海",color);
629     if(strcmp(trainname,"G567")==0)barword24(380,414,BLUE,6,6,"深圳》》广州",color);
630     if(strcmp(trainname,"G751")==0)barword24(380,414,BLUE,6,6,"太原》》北京",color);
631     if(strcmp(trainname,"G768")==0)barword24(380,414,BLUE,6,6,"西安》》太原",color);
632     if(strcmp(trainname,"G267")==0)barword24(380,414,BLUE,6,6,"武汉》》北京",color);
633     if(strcmp(trainname,"G186")==0)barword24(380,414,BLUE,6,6,"重庆》》成都",color);
634     if(strcmp(trainname,"G379")==0)barword24(380,414,BLUE,6,6,"西安》》成都",color);
635     if(strcmp(trainname,"G467")==0)barword24(380,414,BLUE,6,6,"杭州》》上海",color);
636     if(strcmp(trainname,"G685")==0)barword24(380,414,BLUE,6,6,"太原》》西安",color);
637     if(strcmp(trainname,"G335")==0)barword24(380,414,BLUE,6,6,"广州》》深圳",color);
638 }
```

9. page2.c

C

```
1  #include<public.h>
2
3  void pagestart(int *page){
4      int flag=0;                //1为点亮乘客登录, 2为点亮管理员登录
5      clrmos(MouseX,MouseY);
6      delay(100);
7      save_bk_mou(MouseX,MouseY);
8
9      drawstart();
10     while (1)
11     {
12
13         newmouse(&MouseX,&MouseY,&press);
14         if (inbarword32(0,200,BLUE,5,4,"乘客登录",LIGHTGRAY)){
15             if(pressbarword32(0,200,BLUE,5,4,"乘客登录",LIGHTGRAY)==2){
16                 MouseS=1;
17                 if(flag==0){
18                     clrmos(MouseX,MouseY);
19                     delay(10);
20                     barword32(0,200,BLUE,5,4,"乘客登录",RED);
21                     flag=1;
22                 }
23                 continue;
24             }
25             if(pressbarword32(0,200,BLUE,5,4,"乘客登录",LIGHTGRAY)==1){
26                 barword32(0,200,BLUE,5,4,"乘客登录",RED);
27                 MouseS=0;
28                 *page=1;
29                 break;
30             }
31         }
32
33         if (inbarword32(0,300,BLUE,5,5,"管理员登录",LIGHTGRAY)){
34             if(pressbarword32(0,300,BLUE,5,5,"管理员登录",LIGHTGRAY)==2){
35                 MouseS=1;
36                 if (flag==0){
37                     clrmos(MouseX,MouseY);
38                     delay(10);
39                     barword32(0,300,BLUE,5,5,"管理员登录",RED);
40                     flag=2;
41                 }

```



```

42         continue;
43     }
44     if(pressbarword32(0,300,BLUE,5,5,"管理员登录",LIGHTGRAY)==1)
45     {
46         barword32(0,300,BLUE,5,5,"管理员登录",RED);
47         MouseS=0;
48         *page=2;
49         break;
50     }
51 }
52 if(flag){
53     if(flag==1){
54         clrmous(MouseX,MouseY);
55         barword32(0,200,BLUE,5,4,"乘客登录",LIGHTGRAY);
56     }
57     if(flag==2){
58         clrmous(MouseX,MouseY);
59         barword32(0,300,BLUE,5,5,"管理员登录",LIGHTGRAY);
60     }
61     flag=0;
62 }
63 if(MouseS!=0){
64     MouseS=0;
65 }
66 }
67 return;
68 }
69
70 void pagepalogin(int *page,int *pakey){
71     int pos=0;
72     int flag=0;
73     char name[20]={'\0'};
74     char password[20]={'\0'};
75     struct passenger para;
76
77     *pakey=0;
78
79     clrmous(MouseX,MouseY);
80     delay(100);
81     save_bk_mou(MouseX,MouseY);
82
83     drawpalogin();
84
85     while(1){
86         newmouse(&MouseX,&MouseY,&press);
87
88         if(inreturnbutton()==1){
89             if(pressreturnbutton()==2){

```

```

90         MouseS=1;
91         if (flag!=113){
92             clrmouse(MouseX,MouseY);
93             delay(10);
94             returnbutton(RED);
95             flag=113;
96         }
97         continue;
98     }
99     else if(pressreturnbutton()==1){
100         returnbutton(RED);
101         MouseS=0;
102         *page=0;
103         break;
104     }
105 }
106 if(inbarwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED)){
107     if(pressbarwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED)==2){
108         MouseS=2;
109         if(flag==0&&pos==0){
110             flag=1;
111         }
112         continue;
113     }
114     else if(pressbarwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED)==1){
115         MouseS=0;
116         name[0]='\0';
117         barwordframe(180,170,WHITE,11,"",DARKGRAY,RED);
118         Input_Vis(name,180,170,18,WHITE,BLACK);          //可视输入
119         if(strlen(name)!=0){
120             pos=1;
121         }
122         else{
123             pos=0;
124         }
125         continue;
126     }
127 }
128 if(inbarwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED)){
129     if(pressbarwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED)==2){
130         MouseS=2;
131         if(flag==0&&pos==0){
132             flag=1;
133         }
134         continue;
135     }
136     else if(pressbarwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED)==1){
137         MouseS=0;

```

```
137         mouseS=0;
138         password[0]='\0';
139         barwordframe(180,250,WHITE,11,"",DARKGRAY,RED);
140         Input_Invis(password,180,250,18,WHITE,BLACK); //不可视输入
141         if(strlen(password)!=0){
142             pos=1;
143         }
144         else{
145             pos=0;
146         }
147         continue;
148     }
149 }
150 if(inbarword32(0,350,BLUE,3,2,"登录",LIGHTGRAY)){
151     if(pressbarword32(0,350,BLUE,3,2,"登录",LIGHTGRAY)==2){
152         MouseS=1;
153         if (flag==0){
154             clrmous(MouseX,MouseY);
155             delay(10);
156             barword32(0,350,BLUE,3,2,"登录",RED);
157             flag=1;
158         }
159         continue;
160     }
161     if(pressbarword32(0,350,BLUE,3,2,"登录",LIGHTGRAY)==1){
162         MouseS=0;
163         if((*pakey=loginuser(name,password))>0){
164             loginok();
165             delay(300);
166             if(getpassengerbypakey(*pakey,&para)==-1){
167                 *page=4;
168                 break;
169             }
170             *page=5;
171             break;
172         }
173         else if(*pakey==-1){
174             loginpasswordwrong();
175             delay(300);
176             *page=1;
177             break;
178         }
179         else if(*pakey==-2){
180             loginnoregister();
181             delay(300);
182             *page=1;
183             break;
184         }
185     }
```

```
185         else{
186             exit(1);          //todo
187         }
188     }
189 }
190 if(inbarword16(450,420,BLUE,5,5,"新用户注册",LIGHTGRAY)){
191     if(pressbarword16(450,420,BLUE,5,5,"新用户注册",LIGHTGRAY)==2){
192         MouseS=1;
193         if (flag==0){
194             clrmos(MouseX,MouseY);
195             delay(10);
196             barword16(450,420,BLUE,5,5,"新用户注册",RED);
197             flag=5;
198         }
199         continue;
200     }
201     if(pressbarword16(450,420,BLUE,5,5,"新用户注册",LIGHTGRAY)==1){
202         MouseS=0;
203         delay(100);
204         *page=3;
205         break;
206     }
207 }
208 if(flag){
209     if(flag==1){
210         clrmos(MouseX,MouseY);
211         barword32(0,350,BLUE,3,2,"登录",LIGHTGRAY);
212         flag=0;
213     }
214     if(flag==113){
215         clrmos(MouseX,MouseY);
216         returnbutton(LIGHTGRAY);
217         flag=0;
218     }
219     if(flag==5){
220         clrmos(MouseX,MouseY);
221         barword16(450,420,BLUE,5,5,"新用户注册",LIGHTGRAY);
222         flag=0;
223     }
224 }
225 if(MouseS!=0){
226     MouseS=0;
227 }
228 }
229
230 return;
231 }
232
```

```

233 void pageadlogin(int *page){
234     int pos=0;
235     int flag=0;
236     int loginjudge;
237     char name[20]={'\0'};
238     char password[20]={'\0'};
239     clrmous(MouseX,MouseY);
240     delay(100);
241     save_bk_mou(MouseX,MouseY);
242
243     drawadlogin();
244
245     while(1){
246         newmouse(&MouseX,&MouseY,&press);
247         if(inreturnbutton()==1){
248             if(pressreturnbutton()==2){
249                 MouseS=1;
250                 if (flag!=113){
251                     clrmous(MouseX,MouseY);
252                     delay(10);
253                     returnbutton(RED);
254                     flag=113;
255                 }
256                 continue;
257             }
258             else if(pressreturnbutton()==1){
259                 returnbutton(RED);
260                 MouseS=0;
261                 *page=0;
262                 break;
263             }
264         }
265
266         if(inbarwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED)){
267             if(pressbarwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED)==2){
268                 MouseS=2;
269                 if(flag==0&&pos==0){
270                     flag=1;
271                 }
272                 continue;
273             }
274             else if(pressbarwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED)==1){
275                 MouseS=0;
276                 name[0]='\0';
277                 barwordframe(180,170,WHITE,11,"",DARKGRAY,RED);
278                 Input_Vis(name,180,170,18,WHITE,BLACK); //可视输入
279                 if(strlen(name)!=0){

```

```

280         pos=1;
281     }
282     else{
283         pos=0;
284     }
285     continue;
286 }
287 }
288
289 if(inbarwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED)){
290     if(pressbarwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED)==2){
291         MouseS=2;
292         if(flag==0&&pos==0){
293             flag=1;
294         }
295         continue;
296     }
297     else if(pressbarwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED)==1){
298         MouseS=0;
299         password[0]='\0';
300         barwordframe(180,250,WHITE,11,"",DARKGRAY,RED);
301         Input_Invis(password,180,250,18,WHITE,BLACK); //不可视输入
302         if(strlen(password)!=0){
303             pos=1;
304         }
305         else{
306             pos=0;
307         }
308         continue;
309     }
310 }
311
312 if(inbarword32(0,350,BLUE,3,2,"登录",LIGHTGRAY)){
313     if(pressbarword32(0,350,BLUE,3,2,"登录",LIGHTGRAY)==2){
314         MouseS=1;
315         if (flag==0){
316             clrmous(MouseX,MouseY);
317             delay(10);
318             barword32(0,350,BLUE,3,2,"登录",RED);
319             flag=1;
320         }
321         continue;
322     }
323     else if (pressbarword32(0,350,BLUE,3,2,"登录",LIGHTGRAY)==1)
324     {
325         barword32(0,350,BLUE,3,2,"登录",RED);
326         MouseS=0;
327         loginjudge=loginadmin(name,password);

```

```

328         if(loginjudge==1){
329             loginok();
330             delay(300);
331             *page=21;
332             break;
333         }
334         else if(loginjudge==-1){
335             loginpasswordwrong();
336             delay(300);
337             *page=2;
338             break;
339         }
340         else if(loginjudge==-2){
341             loginnoregister();
342             delay(300);
343             *page=2;
344             break;
345         }
346         else{
347             exit(1);           //todo
348         }
349     }
350 }
351
352 if(flag){
353     if(flag==1){
354         clrmous(MouseX,MouseY);
355         barword32(0,350,BLUE,3,2,"登录",LIGHTGRAY);
356         flag=0;
357     }
358     if(flag==113){
359         clrmous(MouseX,MouseY);
360         returnbutton(LIGHTGRAY);
361         flag=0;
362     }
363 }
364 if(MouseS!=0){
365     MouseS=0;
366 }
367 }
368 return;
369 }

```

10. pagead.c

```

1  #include"public.h"
2
3  void pageadmanager(int *page,char searchID[20]){
4      int pos=0;
5      int flag=0;
6      clrmous(MouseX,MouseY);
7      delay(100);
8      save_bk_mou(MouseX,MouseY);
9
10     drawadmanager();
11
12     while(1){
13         newmouse(&MouseX,&MouseY,&press);
14
15         if(inreturnbutton()==1){
16             if(pressreturnbutton()==2){
17                 MouseS=1;
18                 if (flag!=113){
19                     clrmous(MouseX,MouseY);
20                     delay(10);
21                     returnbutton(RED);
22                     flag=113;
23                 }
24                 continue;
25             }
26             else if(pressreturnbutton()==1){
27                 returnbutton(RED);
28                 MouseS=0;
29                 *page=2;
30                 break;
31             }
32         }
33
34         if(inbarwordframe(100,100,WHITE,11,"",WHITE,RED)==1){
35             if(pressbarwordframe(100,100,WHITE,11,"",WHITE,RED)==2){
36                 MouseS=2;
37                 continue;
38             }
39             else if(pressbarwordframe(100,100,WHITE,11,"",WHITE,RED)==1){
40                 MouseS=0;
41                 Input_Vis(searchID,100,100,18,WHITE,BLACK);
42                 if(strlen(searchID)!=0){
43                     pos=1;
44                 }
45                 else{
46                     pos=0;
47                 }

```



```

48         continue;
49     }
50 }
51
52 if(insearchbuttuo() == 1){
53     if(presssearchbutton() == 2){
54         MouseS = 1;
55         if (flag != 822){
56             clrmos(MouseX, MouseY);
57             delay(10);
58             searchbutton(RED);
59             flag = 822;
60         }
61         continue;
62     }
63     else if(presssearchbutton() == 1){
64         returnbutton(RED);
65         MouseS = 0;
66         *page = 215;
67         break;
68     }
69 }
70
71 if (inbarword32(100, 200, CYAN, 6, 6, "标记乘客状况", WHITE)){
72     if(pressbarword32(100, 200, CYAN, 6, 6, "标记乘客状况", WHITE) == 2){
73         MouseS = 1;
74         if(flag == 0){
75             clrmos(MouseX, MouseY);
76             delay(10);
77             barword32(100, 200, CYAN, 6, 6, "标记乘客状况", RED);
78             flag = 1;
79         }
80         continue;
81     }
82     if(pressbarword32(100, 200, CYAN, 6, 6, "标记乘客状况", WHITE) == 1){
83         barword32(100, 200, CYAN, 6, 6, "标记乘客状况", RED);
84         MouseS = 0;
85         *page = 211;
86         break;
87     }
88 }
89
90 if (inbarword32(350, 200, DARKGRAY, 6, 6, "查询阳性乘客", WHITE)){
91     if(pressbarword32(350, 200, DARKGRAY, 6, 6, "查询阳性乘客", WHITE) == 2){
92         MouseS = 1;
93         if(flag == 0){
94             clrmos(MouseX, MouseY);
95             delay(10);

```

```

96         barword32(350,200,DARKGRAY,6,6,"查询阳性乘客",RED);
97         flag=2;
98     }
99     continue;
100 }
101 if(pressbarword32(350,200,DARKGRAY,6,6,"查询阳性乘客",WHITE)==1){
102     barword32(350,200,DARKGRAY,6,6,"查询阳性乘客",RED);
103     MouseS=0;
104     *page=212;
105     break;
106 }
107 }
108
109 if (inbarword32(350,300,BLUE,6,6,"查询密接乘客",WHITE)){
110     if(pressbarword32(350,300,BLUE,6,6,"查询密接乘客",WHITE)==2){
111         MouseS=1;
112         if(flag==0){
113             clrmos(MouseX,MouseY);
114             delay(10);
115             barword32(350,300,BLUE,6,6,"查询密接乘客",RED);
116             flag=3;
117         }
118         continue;
119     }
120     if(pressbarword32(350,300,BLUE,6,6,"查询密接乘客",WHITE)==1){
121         barword32(350,300,BLUE,6,6,"查询密接乘客",RED);
122         MouseS=0;
123         *page=214;
124         break;
125     }
126 }
127
128 if (inbarword32(100,300,BROWN,6,6,"查询可疑车次",WHITE)){
129     if(pressbarword32(100,300,BROWN,6,6,"查询可疑车次",WHITE)==2){
130         MouseS=1;
131         if(flag==0){
132             clrmos(MouseX,MouseY);
133             delay(10);
134             barword32(100,300,BROWN,6,6,"查询可疑车次",RED);
135             flag=4;
136         }
137         continue;
138     }
139     if(pressbarword32(100,300,BROWN,6,6,"查询可疑车次",RED)==1){
140         barword32(100,300,BROWN,6,6,"查询可疑车次",RED);
141         MouseS=0;
142         *page=213;

```

```

143         break;
144     }
145 }
146
147 if(flag){
148     if(flag==1){
149         clrmous(MouseX,MouseY);
150         barword32(100,200,CYAN,6,6,"标记乘客状况",WHITE);
151     }
152     if(flag==2){
153         clrmous(MouseX,MouseY);
154         barword32(350,200,DARKGRAY,6,6,"查询阳性乘客",WHITE);
155     }
156     if(flag==3){
157         clrmous(MouseX,MouseY);
158         barword32(350,300,BLUE,6,6,"查询密接乘客",WHITE);
159     }
160     if(flag==4){
161         clrmous(MouseX,MouseY);
162         barword32(100,300,BROWN,6,6,"查询可疑车次",WHITE);
163     }
164     if(flag==113){
165         clrmous(MouseX,MouseY);
166         returnbutton(LIGHTGRAY);
167     }
168     if(flag==822){
169         clrmous(MouseX,MouseY);
170         searchbutton(WHITE);
171     }
172     flag=0;
173 }
174 if(MouseS!=0){
175     MouseS=0;
176 }
177 }
178 return ;
179 }
180
181 void pageadlabel(int *page){
182     int asymptomatic=0;
183     int symptomatic=0;
184     char ID[20]={'\0'};
185     int trkeyset[50];
186     int pakeyset[50];
187     int counttrkey;
188     int countpakey;
189     int pos=0;
190     int flag=0;

```

```

191     int pakey;
192     int i,j;
193
194     clrmous(MouseX,MouseY);
195     delay(100);
196     save_bk_mou(MouseX,MouseY);
197
198     drawadlabel();
199
200     while(1){
201         newmouse(&MouseX,&MouseY,&press);
202         if(inreturnbutton()==1){
203             if(pressreturnbutton()==2){
204                 MouseS=1;
205                 if (flag!=113){
206                     clrmous(MouseX,MouseY);
207                     delay(10);
208                     returnbutton(RED);
209                     flag=113;
210                 }
211                 continue;
212             }
213             else if(pressreturnbutton()==1){
214                 returnbutton(RED);
215                 MouseS=0;
216                 *page=21;
217                 break;
218             }
219         }
220
221         if(inbarwordframe(230,150,WHITE,11,"",0,RED)){
222             if(pressbarwordframe(230,150,WHITE,11,"",0,RED)==2){
223                 MouseS=2;
224                 continue;
225             }
226             else if(pressbarwordframe(230,150,WHITE,11,"",0,RED)==1){
227                 MouseS=0;
228                 Input_Vis(ID,230,150,18,WHITE,BLACK);
229                 if(strlen(ID)!=0){
230                     pos=1;
231                 }
232                 else{
233                     pos=0;
234                 }
235                 continue;
236             }
237         }
238     }

```

```

230
231
232
233
234
235
236
237
238
239     if(incircle(380,252,10)){
240         if (presscircle(380,252,10)==2)
241             {
242                 MouseS=1;
243                 continue; //todo:预点击?
244             }
245         if (presscircle(380,252,10)==1)
246             {
247                 clrmous(MouseX,MouseY);
248                 setfillstyle(1,BLUE);
249                 setcolor(BLUE);
250
251                 circle(380,252,7);
252                 floodfill(380,252,BLUE);
253                 delay(10);
254                 if(symptomatic==1){
255                     clrmous(MouseX,MouseY);
256                     setfillstyle(1,LIGHTGRAY);
257                     setcolor(LIGHTGRAY);
258
259                     circle(480,252,7);
260                     floodfill(480,252,LIGHTGRAY);
261                     symptomatic=0;
262                 }
263                 asymptomatic=1;
264                 continue;
265             }
266     }
267
268     if(incircle(480,252,10)){
269         if (presscircle(480,252,10)==2)
270             {
271                 MouseS=1;
272                 continue; //todo:预点击?
273             }
274         if (presscircle(480,252,10)==1)
275             {
276                 clrmous(MouseX,MouseY);
277                 setfillstyle(1,BLUE);
278                 setcolor(BLUE);
279
280                 circle(480,252,7);
281                 floodfill(480,252,BLUE);
282                 delay(10);
283                 if(asymptomatic==1){
284                     clrmous(MouseX,MouseY);
285                     setfillstyle(1,LIGHTGRAY);

```

```

286         setcolor(LIGHTGRAY);
287
288         circle(380,252,7);
289         floodfill(380,252,LIGHTGRAY);
290         asymptomatic=0;
291     }
292     symptomatic=1;
293     continue;
294 }
295 }
296
297 if(inbarword32(0,320,GREEN,4,4,"确认录入",WHITE)){
298     if(pressbarword32(0,320,GREEN,4,4,"确认录入",WHITE)==2){
299         MouseS=1;
300         if(flag==0){
301             clrmous(MouseX,MouseY);
302             //todo:???
303             delay(10);
304             barword32(0,320,GREEN,4,4,"确认录入",BLUE);
305             flag=1;
306         }
307         continue;
308     }
309     if(pressbarword32(0,320,GREEN,4,4,"确认录入",WHITE)==1){
310         barword32(0,320,GREEN,4,4,"确认录入",BLUE);
311         MouseS=0;
312         if(getpakeybyID(ID,&pakey)!=1){
313             labelIDzero();
314             delay(300);
315             *page=211;
316             break;;
317         }
318         delay(100);
319         if(symptomatic){
320             updatestatusbypakey(pakey,1);
321             counttrkey=getrecordbypakey(pakey,trkeyset);
322             for(i=0;i<counttrkey;i++){
323                 updatestatusbytrkey(trkeyset[i],1);
324                 countpakey=getrecordbytrkey(trkeyset[i],pakeyset);
325                 for(j=0;j<countpakey;j++){
326                     updatestatusbypakey(pakeyset[j],3);
327                 }
328             }
329         }
330         if(asymptomatic){
331             updatestatusbypakey(pakey,2);
332             counttrkey=getrecordbypakey(pakey,trkeyset);
333             for(i=0;i<counttrkey;i++){

```

```

333         updatestatusbytrkey(trkeyset[i],1);
334         countpakey=getrecordbytrkey(trkeyset[i],pakeyset);
335         for(j=0;j<countpakey;j++){
336             updatestatusbypakey(pakeyset[j],3);
337         }
338     }
339 }
340 labelok();
341 delay(300);
342 *page=211;
343 break;
344 }
345 }
346
347 if(flag){
348     if(flag==1){
349         clrmous(MouseX,MouseY);
350         barword32(0,320,GREEN,4,4,"确认录入",WHITE);
351         flag=0;
352     }
353     if(flag==113){
354         clrmous(MouseX,MouseY);
355         returnbutton(LIGHTGRAY);
356         flag=0;
357     }
358 }
359 if(MouseS!=0){
360     MouseS=0;
361 }
362 }
363 }
364
365 void pageadgetpospa(int *page){
366     int flag;
367     struct passenger target[20];
368     struct passenger *pau;
369     struct passenger *pam;
370     struct pagetrain *pal;
371     int i;
372     int u=0,m=0,l=0;
373
374     int currentpage=1;
375     int countlabel=getpassengerbystatus(1,target);
376     int countpage=(countlabel-1)/3+1;
377
378     FILE *log;
379
380     log=fopen(" \\ LOGFILE" "%d+").

```

```

380     log=fopen("../LOGGER","a+");
381
382     for(i=0;i<countlabel;i++) fprintf(log,"\n %d  %s",target[i].pakey,target
[i].ID);
383
384     fclose(log);
385
386     u=0;
387     m=1;
388     l=2;
389     pau=&target[u];
390     pam=&target[m];
391     pal=&target[l];
392
393
394     if(u>=countlabel){
395         pau=NULL;
396     }
397     if(m>=countlabel){
398         pam=NULL;
399     }
400     if(l>=countlabel){
401         pal=NULL;
402     }
403
404     clrmous(MouseX,MouseY);
405     delay(100);
406     save_bk_mou(MouseX,MouseY);
407
408     drawadgetpospa(pau,pam,pal,currentpage,countpage);
409
410     while(1){
411         newmouse(&MouseX,&MouseY,&press);
412         if(inreturnbutton()==1){
413             if(pressreturnbutton()==2){
414                 MouseS=1;
415                 if (flag!=113){
416                     clrmous(MouseX,MouseY);
417                     delay(10);
418                     returnbutton(RED);
419                     flag=113;
420                 }
421                 continue;
422             }
423             else if(pressreturnbutton()==1){
424                 returnbutton(RED);
425                 MouseS=0;
426                 *page=21;

```



```

427         break;
428     }
429 }
430 if(inleftarrow()==1&&currentpage!=1){
431     if(pressleftarrow()==2){
432         MouseS=1;
433         if (flag!=1){
434             clrmous(MouseX,MouseY);
435             delay(10);
436             leftarrow(RED);
437             flag=1;
438         }
439         continue;
440     }
441     else if(pressleftarrow()==1){
442         MouseS=0;
443         u-=3;
444         m-=3;
445         l-=3;
446         pau=&target[u];
447         pam=&target[m];
448         pal=&target[l];
449         currentpage--;
450         drawadgetpospa(pau,pam,pal,currentpage,countpage);
451         continue;
452     }
453 }
454 if(inrightarrow()==1&&currentpage!=countpage){
455     if(pressrightarrow()==2){
456         MouseS=1;
457         if (flag!=2){
458             clrmous(MouseX,MouseY);
459             delay(10);
460             rightarrow(RED);
461             flag=2;
462         }
463         continue;
464     }
465     else if(pressrightarrow()==1){
466         MouseS=0;
467         u+=3;
468         m+=3;
469         l+=3;
470         if(u>=countlabel){
471             pau=NULL;
472         }
473         else{
474             pau=&target[u];

```

```

475         }
476         if(m>=countlabel){
477             pam=NULL;
478         }
479         else{
480             pam=&target[m];
481         }
482         if(l>=countlabel){
483             pal=NULL;
484         }
485         else{
486             pal=&target[u];
487         }
488         currentpage++;
489         drawadgetpospa(pau,pam,pal,currentpage,countpage);
490         continue;
491     }
492 }
493 if(flag){
494     if(flag==113){
495         clrmous(MouseX,MouseY);
496         returnbutton(LIGHTGRAY);
497         flag=0;
498     }
499     if(flag==1){
500         clrmous(MouseX,MouseY);
501         leftarrow(BLUE);
502         flag=0;
503     }
504     if(flag==2){
505         clrmous(MouseX,MouseY);
506         rightarrow(BLUE);
507         flag=0;
508     }
509 }
510 if(MouseS!=0){
511     MouseS=0;
512 }
513 }
514 }
515
516 void pageadgetcttpa(int *page){
517     int flag;
518     struct passenger target[20];
519     struct passenger *pau;
520     struct passenger *pam;
521     struct pagetrain *pal;
522     int i;

```

```

522     int i;
523     int u=0,m=0,l=0;
524
525     int currentpage=1;
526     int countlabel=getpassengerbystatus(3,target);
527     int countpage=(countlabel-1)/3+1;
528
529     u=0;
530     m=1;
531     l=2;
532     pau=&target[u];
533     pam=&target[m];
534     pal=&target[l];
535
536     if(u>=countlabel){
537         pau=NULL;
538     }
539     if(m>=countlabel){
540         pam=NULL;
541     }
542     if(l>=countlabel){
543         pal=NULL;
544     }
545
546     clrmous(MouseX,MouseY);
547     delay(100);
548     save_bk_mou(MouseX,MouseY);
549
550     drawadgetcttpa(pau,pam,pal,currentpage,countpage);
551
552     while(1){
553         newmouse(&MouseX,&MouseY,&press);
554         if(inreturnbutton()==1){
555             if(pressreturnbutton()==2){
556                 MouseS=1;
557                 if (flag!=113){
558                     clrmous(MouseX,MouseY);
559                     delay(10);
560                     returnbutton(RED);
561                     flag=113;
562                 }
563                 continue;
564             }
565             else if(pressreturnbutton()==1){
566                 returnbutton(RED);
567                 MouseS=0;
568                 *page=21;
569                 break;

```

```

570     }
571 }
572 if(inleftarrow()==1&&currentpage!=1){
573     if(pressleftarrow()==2){
574         MouseS=1;
575         if (flag!=1){
576             clrmous(MouseX,MouseY);
577             delay(10);
578             leftarrow(RED);
579             flag=1;
580         }
581         continue;
582     }
583     else if(pressleftarrow()==1){
584         MouseS=0;
585         u-=3;
586         m-=3;
587         l-=3;
588         pau=&target[u];
589         pam=&target[m];
590         pal=&target[l];
591         currentpage--;
592         drawadgetcttpa(pau,pam,pal,currentpage,countpage);
593         continue;
594     }
595 }
596 if(inrightarrow()==1&&currentpage!=countpage){
597     if(pressrightarrow()==2){
598         MouseS=1;
599         if (flag!=2){
600             clrmous(MouseX,MouseY);
601             delay(10);
602             rightarrow(RED);
603             flag=2;
604         }
605         continue;
606     }
607     else if(pressrightarrow()==1){
608         MouseS=0;
609         u+=3;
610         m+=3;
611         l+=3;
612         if(u>=countlabel){
613             pau=NULL;
614         }
615         else{
616             pau=&target[u];
617         }

```

```

618         if(m>=countlabel){
619             pam=NULL;
620         }
621         else{
622             pam=&target[m];
623         }
624         if(l>=countlabel){
625             pal=NULL;
626         }
627         else{
628             pal=&target[u];
629         }
630         currentpage++;
631         drawadgetcttpa(pau,pam,pal,currentpage,countpage);
632         continue;
633     }
634 }
635 if(flag){
636     if(flag==113){
637         clrmous(MouseX,MouseY);
638         returnbutton(LIGHTGRAY);
639         flag=0;
640     }
641     if(flag==1){
642         clrmous(MouseX,MouseY);
643         leftarrow(BLUE);
644         flag=0;
645     }
646     if(flag==2){
647         clrmous(MouseX,MouseY);
648         rightarrow(BLUE);
649         flag=0;
650     }
651 }
652 if(MouseS!=0){
653     MouseS=0;
654 }
655 }
656 }
657
658 void pageadgetpostrn(int *page){
659     int flag;
660     struct pagetrain target[20];
661     struct pagetrain *ptrnu;
662     struct pagetrain *ptrnm;
663     struct pagetrain *ptrnl;
664     int i;

```

```
665 //todo u的值
666 int u=0,m=0,l=0;
667
668 int currentpage=1;
669 int countlabel=gettrainbystatus(1,target);
670 int countpage=(countlabel-1)/3+1;
671
672 u=0;
673 m=1;
674 l=2;
675 ptrnu=&target[u];
676 ptrnm=&target[m];
677 ptrnl=&target[l];
678
679 if(u>=countlabel){
680     ptrnu=NULL;
681 }
682 if(m>=countlabel){
683     ptrnm=NULL;
684 }
685 if(l>=countlabel){
686     ptrnl=NULL;
687 }
688
689 clrmous(MouseX,MouseY);
690 delay(100);
691 save_bk_mou(MouseX,MouseY);
692
693 drawadgetpostr(ptrnu,ptrnm,ptrnl,currentpage,countpage);
694
695 while(1){
696     newmouse(&MouseX,&MouseY,&press);
697     if(inreturnbutton()==1){
698         if(pressreturnbutton()==2){
699             MouseS=1;
700             if (flag!=113){
701                 clrmous(MouseX,MouseY);
702                 delay(10);
703                 returnbutton(RED);
704                 flag=113;
705             }
706             continue;
707         }
708         else if(pressreturnbutton()==1){
709             returnbutton(RED);
710             MouseS=0;
711             *page=21;
712             break;
```

```

713     }
714 }
715 if(inleftarrow()==1&&currentpage!=1){
716     if(pressleftarrow()==2){
717         MouseS=1;
718         if (flag!=1){
719             clrmous(MouseX,MouseY);
720             delay(10);
721             leftarrow(RED);
722             flag=1;
723         }
724         continue;
725     }
726     else if(pressleftarrow()==1){
727         MouseS=0;
728         u-=3;
729         m-=3;
730         l-=3;
731         ptrnu=&target[u];
732         ptrnm=&target[m];
733         ptrnl=&target[l];
734         currentpage--;
735         drawadgetpostr(ptrnu,ptrnm,ptrnl,currentpage,countpage);
736         continue;
737     }
738 }
739 if(inrightarrow()==1&&currentpage!=countpage){
740     if(pressrightarrow()==2){
741         MouseS=1;
742         if (flag!=2){
743             clrmous(MouseX,MouseY);
744             delay(10);
745             rightarrow(RED);
746             flag=2;
747         }
748         continue;
749     }
750     else if(pressrightarrow()==1){
751         MouseS=0;
752         u+=3;
753         m+=3;
754         l+=3;
755         if(u>=countlabel){
756             ptrnu=NULL;
757         }
758         else{
759             ptrnu=&target[u];
760         }

```

```

760         }
761         if(m>=countlabel){
762             ptrnm=NULL;
763         }
764         else{
765             ptrnm=&target[m];
766         }
767         if(l>=countlabel){
768             ptrnl=NULL;
769         }
770         else{
771             ptrnl=&target[u];
772         }
773         currentpage++;
774         drawadgetpostr(ptrnu,ptrnm,ptrnl,currentpage,countpage);
775         continue;
776     }
777 }
778 if(flag){
779     if(flag==113){
780         clrmos(MouseX,MouseY);
781         returnbutton(LIGHTGRAY);
782         flag=0;
783     }
784     if(flag==1){
785         clrmos(MouseX,MouseY);
786         leftarrow(BLUE);
787         flag=0;
788     }
789     if(flag==2){
790         clrmos(MouseX,MouseY);
791         rightarrow(BLUE);
792         flag=0;
793     }
794 }
795 if(MouseS!=0){
796     MouseS=0;
797 }
798 }
799 }
800
801 void pageadsearch(int *page,char searchID[20],int *pakey){
802     int flag;
803     struct passenger target[20];
804     struct passenger *pau;
805     struct passenger *pam;
806     struct passenger *pal;
807     int i;

```



```
808 //todo u的值
809 int u=0,m=0,l=0;
810
811 int currentpage=1;
812 int countlabel=getpassengerbysearch(searchID,target);
813 int countpage=(countlabel-1)/3+1;
814
815 memset(searchID,'\0',sizeof(searchID));
816
817 u=0;
818 m=1;
819 l=2;
820 pau=&target[u];
821 pam=&target[m];
822 pal=&target[l];
823
824 if(u>=countlabel){
825     pau=NULL;
826 }
827 if(m>=countlabel){
828     pam=NULL;
829 }
830 if(l>=countlabel){
831     pal=NULL;
832 }
833
834 clrmous(MouseX,MouseY);
835 delay(100);
836 save_bk_mou(MouseX,MouseY);
837
838 drawadsearch(pau,pam,pal,currentpage,countpage);
839
840 while(1){
841     newmouse(&MouseX,&MouseY,&press);
842     if(inreturnbutton()==1){
843         if(pressreturnbutton()==2){
844             MouseS=1;
845             if (flag!=113){
846                 clrmous(MouseX,MouseY);
847                 delay(10);
848                 returnbutton(RED);
849                 flag=113;
850             }
851             continue;
852         }
853         else if(pressreturnbutton()==1){
854             returnbutton(RED);
855             MouseS=0;
```

```

856         *page=21;
857         break;
858     }
859 }
860 if(inleftarrow()==1&&currentpage!=1){
861     if(pressleftarrow()==2){
862         MouseS=1;
863         if (flag!=1){
864             clrmous(MouseX,MouseY);
865             delay(10);
866             leftarrow(RED);
867             flag=1;
868         }
869         continue;
870     }
871     else if(pressleftarrow()==1){
872         MouseS=0;
873         u-=3;
874         m-=3;
875         l-=3;
876         pau=&target[u];
877         pam=&target[m];
878         pal=&target[u];
879         currentpage--;
880         drawadsearch(pau,pam,pal,currentpage,countpage);
881         continue;
882     }
883 }
884 if(inrightarrow()==1&&currentpage!=countpage){
885     if(pressrightarrow()==2){
886         MouseS=1;
887         if (flag!=2){
888             clrmous(MouseX,MouseY);
889             delay(10);
890             rightarrow(RED);
891             flag=2;
892         }
893         continue;
894     }
895     else if(pressrightarrow()==1){
896         MouseS=0;
897         u+=3;
898         m+=3;
899         l+=3;
900         if(u>=countlabel){
901             pau=NULL;
902         }
903     }
904     else if(pressrightarrow()==2){

```

```

903         else{
904             pau=&target[u];
905         }
906         if(m>=countlabel){
907             pam=NULL;
908         }
909         else{
910             pam=&target[m];
911         }
912         if(l>=countlabel){
913             pal=NULL;
914         }
915         else{
916             pal=&target[u];
917         }
918         currentpage++;
919         drawadsearch(pau,pam,pal,currentpage,countpage);
920         continue;
921     }
922 }
923 if(u<countlabel&&inpalabelu()==1){
924     if(presspalabelu()==2){
925         MouseS=1;
926         continue;
927     }
928     if(presspalabelu()==1){
929         *pakey=pau->pakey;
930         delay(10);
931         *page=221;
932         break;
933     }
934 }
935 if(m<countlabel&&inpalabelm()==1){
936     if(presspalabelm()==2){
937         MouseS=1;
938         continue;
939     }
940     if(presspalabelm()==1){
941         *pakey=pam->pakey;
942         delay(10);
943         *page=221;
944         break;
945     }
946 }
947 if(l<countlabel&&inpalabell()==1){
948     if(presspalabell()==2){
949         MouseS=1;
950         continue;

```

```

951         }
952         if(presspalabell()==1){
953             *pakey=pal->pakey;
954             delay(10);
955             *page=221;
956             break;
957         }
958     }
959     if(flag){
960         if(flag==113){
961             clrmos(MouseX,MouseY);
962             returnbutton(LIGHTGRAY);
963             flag=0;
964         }
965         if(flag==1){
966             clrmos(MouseX,MouseY);
967             leftarrow(BLUE);
968             flag=0;
969         }
970         if(flag==2){
971             clrmos(MouseX,MouseY);
972             rightarrow(BLUE);
973             flag=0;
974         }
975     }
976     if(MouseS!=0){
977         MouseS=0;
978     }
979 }
980 }
981
982 void pageadpamessage(int *page,int *pakey){
983     int flag=0;
984     int pos=0;
985     int judgepakey;
986     struct passenger pa;
987
988     //todo
989     judgepakey=getpassengerbypakey(*pakey,&pa);
990
991     clrmos(MouseX,MouseY);
992     delay(100);
993     save_bk_mou(MouseX,MouseY);
994
995     drawadpamessage(&pa);
996
997     while(1){
998         newmouse(&MouseX,&MouseY,&press);

```

```

999         if(inreturnbutton()==1){
1000             if(pressreturnbutton()==2){
1001                 MouseS=1;
1002                 if (flag!=113){
1003                     clrmos(MouseX,MouseY);
1004                     delay(10);
1005                     returnbutton(RED);
1006                     flag=113;
1007                 }
1008                 continue;
1009             }
1010             else if(pressreturnbutton()==1){
1011                 returnbutton(RED);
1012                 MouseS=0;
1013                 *page=21;
1014                 break;
1015             }
1016         }
1017         if(inbarword32(265,400,BLUE,4,4,"乘车记录",LIGHTGRAY)){
1018             if(pressbarword32(265,400,BLUE,4,4,"乘车记录",LIGHTGRAY)==2){
1019                 MouseS=1;
1020                 if(flag!=2){
1021                     clrmos(MouseX,MouseY);
1022                     delay(10);
1023                     barword32(265,400,BLUE,4,4,"乘车记录",RED);
1024                     flag=2;
1025                 }
1026                 continue;
1027             }
1028             else if(pressbarword32(265,400,BLUE,4,4,"乘车记录",LIGHTGRAY)==1){
1029                 MouseS=0;
1030                 delay(100);
1031                 *page=22;
1032                 break;
1033             }
1034         }
1035         if(inbarword32(465,400,BLUE,4,4,"轨迹查询",LIGHTGRAY)){
1036             if(pressbarword32(465,400,BLUE,4,4,"轨迹查询",LIGHTGRAY)==2){
1037                 MouseS=1;
1038                 if(flag!=9){
1039                     clrmos(MouseX,MouseY);
1040                     delay(10);
1041                     barword32(465,400,BLUE,4,4,"轨迹查询",RED);
1042                     flag=9;
1043                 }
1044                 continue;
1045             }

```

```

1046         else if (pressbarword32(465,400,BLUE,4,4,"轨迹查询",LIGHTGRAY)==1){
1047             MouseS=0;
1048             delay(100);
1049             *page=223;
1050             break;
1051         }
1052     }
1053     if(flag){
1054         if(flag==2){
1055             clrmous(MouseX,MouseY);
1056             barword32(265,400,BLUE,4,4,"乘车记录",LIGHTGRAY);
1057             flag=0;
1058         }
1059         if(flag==9){
1060             clrmous(MouseX,MouseY);
1061             barword32(465,400,BLUE,4,4,"轨迹查询",LIGHTGRAY);
1062             flag=0;
1063         }
1064         if(flag==113){
1065             clrmous(MouseX,MouseY);
1066             returnbutton(LIGHTGRAY);
1067             flag=0;
1068         }
1069     }
1070     if(MouseS!=0){
1071         MouseS=0;
1072     }
1073
1074 }
1075
1076 }
1077
1078 void pageadpagetrecord(int *page,int *pakey){
1079     int flag;
1080     int trkeyset[20]={0};
1081     struct train *ptrnu;
1082     struct train *ptrnm;
1083     struct train *ptrnl;
1084     int i;
1085
1086     int currentpage=1;
1087     int countlabel=getrecordbypakey(*pakey,trkeyset);
1088     int countpage=(countlabel-1)/3+1;
1089
1090     int u=0,m=0,l=0;
1091
1092     ptrnu=(struct train *)malloc(sizeof(TRAIN));
1093     ptrnm=(struct train *)malloc(sizeof(TRAIN));

```

```

1094     ptrnl=(struct train *)malloc(sizeof(TRAIN));
1095
1096     u=0;
1097     m=1;
1098     l=2;
1099     gettrainbytrkey(trkeyset[u],ptrnu);
1100     gettrainbytrkey(trkeyset[m],ptrnm);
1101     gettrainbytrkey(trkeyset[l],ptrnl);
1102
1103     if(u>=countlabel){
1104         ptrnu=NULL;
1105     }
1106     if(m>=countlabel){
1107         ptrnm=NULL;
1108     }
1109     if(l>=countlabel){
1110         ptrnl=NULL;
1111     }
1112
1113     clrmous(MouseX,MouseY);
1114     delay(100);
1115     save_bk_mou(MouseX,MouseY);
1116
1117     drawpagetrecord(ptrnu,ptrnm,ptrnl,currentpage,countpage);
1118
1119     while(1){
1120         newmouse(&MouseX,&MouseY,&press);
1121         if(inreturnbutton()==1){
1122             if(pressreturnbutton()==2){
1123                 MouseS=1;
1124                 if (flag!=113){
1125                     clrmous(MouseX,MouseY);
1126                     delay(10);
1127                     returnbutton(RED);
1128                     flag=113;
1129                 }
1130                 continue;
1131             }
1132             else if(pressreturnbutton()==1){
1133                 returnbutton(RED);
1134                 MouseS=0;
1135                 *page=221;
1136                 free(ptrnu);
1137                 free(ptrnm);
1138                 free(ptrnl);
1139                 break;
1140             }
1141         }

```

```

1142     if(inleftarrow()==1&&currentpage!=1){
1143         if(pressleftarrow()==2){
1144             MouseS=1;
1145             if (flag!=1){
1146                 clrmous(MouseX,MouseY);
1147                 delay(10);
1148                 leftarrow(RED);
1149                 flag=1;
1150             }
1151             continue;
1152         }
1153     else if(pressleftarrow()==1){
1154         MouseS=0;
1155         u-=3;
1156         m-=3;
1157         l-=3;
1158         free(ptrnu);
1159         free(ptrnm);
1160         free(ptrnl);
1161         ptrnu=(struct train *)malloc(sizeof(TRAIN));
1162         ptrnm=(struct train *)malloc(sizeof(TRAIN));
1163         ptrnl=(struct train *)malloc(sizeof(TRAIN));
1164         gettrainbytrkey(trkeyset[u],ptrnu);
1165         gettrainbytrkey(trkeyset[m],ptrnm);
1166         gettrainbytrkey(trkeyset[l],ptrnl);
1167         currentpage--;
1168         drawpagetrecord(ptrnu,ptrnm,ptrnl,currentpage,countpage);
1169         continue;
1170     }
1171 }
1172 if(inrightarrow()==1&&currentpage!=countpage){
1173     if(pressrightarrow()==2){
1174         MouseS=1;
1175         if (flag!=2){
1176             clrmous(MouseX,MouseY);
1177             delay(10);
1178             rightarrow(RED);
1179             flag=2;
1180         }
1181         continue;
1182     }
1183 else if(pressrightarrow()==1){
1184     MouseS=0;
1185     u+=3;
1186     m+=3;
1187     l+=3;
1188     free(ptrnu);

```



```

1189         free(ptrnm);
1190         free(ptrnl);
1191         ptrnu=(struct train *)malloc(sizeof(TRAIN));
1192         ptrnm=(struct train *)malloc(sizeof(TRAIN));
1193         ptrnl=(struct train *)malloc(sizeof(TRAIN));
1194         gettrainbytrkey(trkeyset[u],ptrnu);
1195         gettrainbytrkey(trkeyset[m],ptrnm);
1196         gettrainbytrkey(trkeyset[l],ptrnl);
1197         if(u>=countlabel){
1198             ptrnu=NULL;
1199         }
1200         if(m>=countlabel){
1201             ptrnm=NULL;
1202         }
1203         if(l>=countlabel){
1204             ptrnl=NULL;
1205         }
1206         currentpage++;
1207         drawpagetrecord(ptrnu,ptrnm,ptrnl,currentpage,countpage);
1208         continue;
1209     }
1210 }
1211 if(flag){
1212     if(flag==113){
1213         clrmous(MouseX,MouseY);
1214         returnbutton(LIGHTGRAY);
1215         flag=0;
1216     }
1217     if(flag==1){
1218         clrmous(MouseX,MouseY);
1219         leftarrow(BLUE);
1220         flag=0;
1221     }
1222     if(flag==2){
1223         clrmous(MouseX,MouseY);
1224         rightarrow(BLUE);
1225         flag=0;
1226     }
1227 }
1228 if(MouseS!=0){
1229     MouseS=0;
1230 }
1231 }
1232 }
1233
1234 void pageadpagetrack(int *page,int *pakey){
1235     int flag;
1236     char trackset[10][10];

```

```

1237     int i;
1238     int trackcount=gettrackbypakey(*pakey,trackset);
1239
1240     clrmous(MouseX,MouseY);
1241     delay(100);
1242     save_bk_mou(MouseX,MouseY);
1243
1244     drawpagetrack();
1245
1246     for(i=0;i<trackcount;i++){
1247         mapline(trackset[i],RED);
1248     }
1249
1250     while(1){
1251         newmouse(&MouseX,&MouseY,&press);
1252         if(inreturnbutton()==1){
1253             if(pressreturnbutton()==2){
1254                 MouseS=1;
1255                 if (flag!=113){
1256                     clrmous(MouseX,MouseY);
1257                     delay(10);
1258                     returnbutton(RED);
1259                     flag=113;
1260                 }
1261                 continue;
1262             }
1263             else if(pressreturnbutton()==1){
1264                 returnbutton(RED);
1265                 MouseS=0;
1266                 *page=221;
1267                 break;
1268             }
1269         }
1270         if(flag){
1271             if(flag==113){
1272                 clrmous(MouseX,MouseY);
1273                 returnbutton(LIGHTGRAY);
1274                 flag=0;
1275             }
1276         }
1277         if(MouseS!=0){
1278             MouseS=0;
1279         }
1280     }
1281 }
1282

```

11. pagepa.c

JavaScript

```
1  #include "public.h"
2
3  void pageparegister(int *page){
4      int pos=0;
5      int flag=0;
6      char name[20]={'\0'};
7      char password1[20]={'\0'};
8      char password2[20]={'\0'};
9
10     clrmous(MouseX,MouseY);
11     delay(100);
12     save_bk_mou(MouseX,MouseY);
13
14     drawparegister();
15
16     while(1){
17         newmouse(&MouseX,&MouseY,&press);
18
19         if(inreturnbutton()==1){
20             if(pressreturnbutton()==2){
21                 MouseS=1;
22                 if (flag!=113){
23                     clrmous(MouseX,MouseY);
24                     delay(10);
25                     returnbutton(RED);
26                     flag=113;
27                 }
28                 continue;
29             }
30             else if(pressreturnbutton()==1){
31                 returnbutton(RED);
32                 MouseS=0;
33                 *page=0;
34                 break;
35             }
36         }
37         if(inbarwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED)){
38             if(pressbarwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED)==2){
39                 MouseS=2;
40                 if(flag==0&&pos==0){
41                     flag=1;
42                 }
43                 continue;
```

```

44     }
45     else if(pressbarwordframe(180,170,WHITE,11,"",LIGHTGRAY,RED)==1){
46         MouseS=0;
47         name[0]='\0';
48         barwordframe(180,170,WHITE,11,"",DARKGRAY,RED);
49         Input_Vis(name,180,170,18,WHITE,BLACK); //可视输入
50         if(strlen(name)!=0){
51             pos=1;
52         }
53         else{
54             pos=0;
55         }
56         continue;
57     }
58 }
59 if(inbarwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED)){
60     if(pressbarwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED)==2){
61         MouseS=2;
62         if(flag==0&&pos==0){
63             flag=1;
64         }
65         continue;
66     }
67     else if(pressbarwordframe(180,250,WHITE,11,"",LIGHTGRAY,RED)==1){
68         MouseS=0;
69         password1[0]='\0';
70         barwordframe(180,250,WHITE,11,"",DARKGRAY,RED);
71         Input_Invis(password1,180,250,18,WHITE,BLACK); //不可视输入
72         if(strlen(password1)!=0){
73             pos=1;
74         }
75         else{
76             pos=0;
77         }
78         continue;
79     }
80 }
81 if(inbarwordframe(180,330,WHITE,11,"",LIGHTGRAY,RED)){
82     if(pressbarwordframe(180,330,WHITE,11,"",LIGHTGRAY,RED)==2){
83         MouseS=2;
84         if(flag==0&&pos){
85             flag=1;
86         }
87         continue;
88     }
89     else if(pressbarwordframe(180,330,WHITE,11,"",LIGHTGRAY,RED)==1){
90         MouseS=0;
91         password2[0]='\0';

```

```

92         delay(10);
93         barwordframe(180,330,WHITE,11,"",LIGHTGRAY,RED);
94         barwordframe(180,330,WHITE,11,"",LIGHTGRAY,RED);
95         Input_Invis(password2,180,330,18,WHITE,BLACK);
96         if(strlen(password2)==0){
97             barwordframe(180,330,WHITE,11,"",LIGHTGRAY,RED);
98             puthz(190,335,"再次输入密码",32,32,LIGHTGRAY); //todo
99         }
100        if(strlen(password1)!=0){
101            pos=1;
102        }
103        else{
104            pos=0;
105        }
106        continue;
107    }
108 }
109 if(inbarword32(0,400,BLUE,2,2,"注册",LIGHTGRAY)){
110     if(pressbarword32(0,400,BLUE,2,2,"注册",LIGHTGRAY)==2){
111         MouseS=1;
112         if (flag==0){
113             clrmos(MouseX,MouseY);
114             delay(10);
115             barword32(0,400,BLUE,2,2,"注册",RED);
116             flag=1;
117         }
118         continue;
119     }
120     if(pressbarword32(0,400,BLUE,2,2,"注册",LIGHTGRAY)==1){
121         MouseS=0;
122         if(strlen(name)==0){
123             registernamezero();
124             delay(1000);
125             *page=3;
126             break;
127         }
128         if(strlen(password1)==0){
129             registerpasszero();
130             delay(1000);
131             *page=3;
132             break;
133         }
134         if(strcmp(password1,password2)!=0){
135             registerpasswrong();
136             delay(1000);
137             *page=3;
138             break;
139         }

```

```

140         if(strlen(password1)<6){
141             registerpassshort();
142             delay(1000);
143             *page=3;
144             break;
145         }
146         if(existusername(name)==1){
147             registerrepeat();
148             delay(1000);
149             *page=3;
150             break;
151         }
152         else{
153             writeuser(name,password1);
154             registerok();
155             delay(1000);
156             *page=1;
157             break;
158         }
159     }
160 }
161 if(flag){
162     if(flag==1){
163         clrmos(MouseX,MouseY);
164         barword32(0,400,BLUE,2,2,"注册",LIGHTGRAY);
165         flag=0;
166     }
167     if(flag==113){
168         clrmos(MouseX,MouseY);
169         returnbutton(LIGHTGRAY);
170         flag=0;
171     }
172 }
173 if(MouseS!=0){
174     MouseS=0;
175 }
176 }
177 }
178
179 void pagepabind(int *page,int *pakey){
180     int pos=0;
181     int flag=0;
182     char tel[20]={'\0'};
183     char ID[20]={'\0'};
184
185     clrmos(MouseX,MouseY);
186     delay(100);

```

```

187     save_bk_mou(MouseX,MouseY);
188
189     drawpabind();
190
191     while(1){
192         newmouse(&MouseX,&MouseY,&press);
193
194         if(inreturnbutton()==1){
195             if(pressreturnbutton()==2){
196                 MouseS=1;
197                 if (flag!=113){
198                     clrmous(MouseX,MouseY);
199                     delay(10);
200                     returnbutton(RED);
201                     flag=113;
202                 }
203                 continue;
204             }
205             else if(pressreturnbutton()==1){
206                 returnbutton(RED);
207                 MouseS=0;
208                 *page=1;
209                 break;
210             }
211         }
212
213         if(inbarwordframe(210,200,WHITE,11,"",LIGHTGRAY,RED)){
214             if(pressbarwordframe(210,200,WHITE,11,"",LIGHTGRAY,RED)==2){
215                 MouseS=2;
216                 if(flag==0&&pos==0){
217                     flag=1;
218                 }
219                 continue;
220             }
221             else if(pressbarwordframe(210,200,WHITE,11,"",LIGHTGRAY,RED)==1){
222                 MouseS=0;
223                 ID[0]='\0';
224                 barwordframe(210,200,WHITE,11,"",LIGHTGRAY,RED);
225                 Input_Vis(ID,210,200,18,WHITE,BLACK);
226                 if(strlen(ID)!=0){
227                     pos=1;
228                 }
229                 else{
230                     pos=0;
231                 }
232                 continue;
233             }
234         }

```

```

235     if(inbarwordframe(210,300,WHITE,11,"",LIGHTGRAY,RED)){
236         if(pressbarwordframe(210,300,WHITE,11,"",LIGHTGRAY,RED)==2){
237             MouseS=2;
238             if(flag==0&&pos==0){
239                 flag=1;
240             }
241             continue;
242         }
243         else if(pressbarwordframe(210,300,WHITE,11,"",LIGHTGRAY,RED)==1){
244             MouseS=0;
245             tel[0]='\0';
246             barwordframe(210,300,WHITE,11,"",LIGHTGRAY,RED);
247             Input_Vis(tel,210,300,11,WHITE,BLACK);
248             if(strlen(tel)!=0){
249                 pos=1;
250             }
251             else{
252                 pos=0;
253             }
254             continue;
255         }
256     }
257     if(inbarword32(0,400,BLUE,2,2,"绑定",WHITE)){
258         if(pressbarword32(0,400,BLUE,2,2,"绑定",WHITE)==2){
259             MouseS=1;
260             if (flag!=5){
261                 clrmous(MouseX,MouseY);
262                 delay(10);
263                 barword32(0,400,BLUE,2,2,"绑定",RED);
264                 flag=5;
265             }
266             continue;
267         }
268         else if(pressbarword32(0,400,BLUE,2,2,"绑定",WHITE)==1){
269             MouseS=0;
270             if(judgeID(ID)!=1){
271                 bindIDwrong();
272                 delay(300);
273                 *page=4;
274                 break;
275             }
276             if(judgetel(tel)!=1){
277                 bindtelwrong();
278                 delay(300);
279                 *page=4;
280                 break;
281             }
282             if(existpassengerID(ID)--1){

```



```

282         if (existpassengerID(ID)--1){
283             bindIDrepeat();
284             delay(300);
285             *page=4;
286             break;
287         }
288         writepassenger(*pakey,ID,tel,getsexbyID(ID),getagebyID(ID),0);
289         bindok();
290         delay(1000);
291         *page=5;
292         break;
293     }
294 }
295 if(flag){
296     if(flag==5){
297         clrmous(MouseX,MouseY);
298         barword32(0,400,BLUE,2,2,"绑定",WHITE);
299         flag=0;
300     }
301     if(flag==113){
302         clrmous(MouseX,MouseY);
303         returnbutton(LIGHTGRAY);
304         flag=0;
305     }
306 }
307 if(MouseS!=0){
308     MouseS=0;
309 }
310
311 }
312 }
313
314 void pagepamessage(int *page,int *pakey){
315     int flag=0;
316     int pos=0;
317     int judgepakey;
318     struct passenger pa;
319
320     //todo
321     judgepakey=getpassengerbypakey(*pakey,&pa);
322
323     clrmous(MouseX,MouseY);
324     delay(100);
325     save_bk_mou(MouseX,MouseY);
326
327     drawpamessage(&pa);
328
329     while(1){

```

```
330     newmouse(&MouseX,&MouseY,&press);
331     if(inreturnbutton()==1){
332         if(pressreturnbutton()==2){
333             MouseS=1;
334             if (flag!=113){
335                 clrmos(MouseX,MouseY);
336                 delay(10);
337                 returnbutton(RED);
338                 flag=113;
339             }
340             continue;
341         }
342         else if(pressreturnbutton()==1){
343             returnbutton(RED);
344             MouseS=0;
345             *page=0;
346             break;
347         }
348     }
349     if(inbarword32(65,400,BLUE,4,4,"登记乘车",LIGHTGRAY)){
350         if(pressbarword32(65,400,BLUE,4,4,"登记乘车",LIGHTGRAY)==2){
351             MouseS=1;
352             if(flag!=1){
353                 clrmos(MouseX,MouseY);
354                 delay(10);
355                 barword32(65,400,BLUE,4,4,"登记乘车",RED);
356                 flag=1;
357             }
358             continue;
359         }
360         else if(pressbarword32(65,400,BLUE,4,4,"登记乘车",LIGHTGRAY)==1){
361             if(pa.status!=0){
362                 statusabnormal();
363                 delay(300);
364                 *page=5;
365                 break;;
366             }
367             MouseS=0;
368             delay(100);
369             *page=6;
370             break;
371         }
372     }
373     if(inbarword32(265,400,BLUE,4,4,"乘车记录",LIGHTGRAY)){
374         if(pressbarword32(265,400,BLUE,4,4,"乘车记录",LIGHTGRAY)==2){
375             MouseS=1;
376             if(flag!=2){
377                 clrmos(MouseX,MouseY);
```

```

378         delay(10);
379         barword32(265,400,BLUE,4,4,"乘车记录",RED);
380         flag=2;
381     }
382     continue;
383 }
384 else if(pressbarword32(265,400,BLUE,4,4,"乘车记录",LIGHTGRAY)==1){
385     MouseS=0;
386     delay(100);
387     *page=7;
388     break;
389 }
390 }
391 if(inbarword32(465,400,BLUE,4,4,"轨迹查询",LIGHTGRAY)){
392     if(pressbarword32(465,400,BLUE,4,4,"轨迹查询",LIGHTGRAY)==2){
393         MouseS=1;
394         if(flag!=9){
395             clrmos(MouseX,MouseY);
396             delay(10);
397             barword32(465,400,BLUE,4,4,"轨迹查询",RED);
398             flag=9;
399         }
400         continue;
401     }
402     else if(pressbarword32(465,400,BLUE,4,4,"轨迹查询",LIGHTGRAY)==1){
403         MouseS=0;
404         delay(100);
405         *page=9;
406         break;
407     }
408 }
409 if(flag){
410     if(flag==1){
411         clrmos(MouseX,MouseY);
412         barword32(65,400,BLUE,4,4,"登记乘车",LIGHTGRAY);
413         flag=0;
414     }
415     if(flag==2){
416         clrmos(MouseX,MouseY);
417         barword32(265,400,BLUE,4,4,"乘车记录",LIGHTGRAY);
418         flag=0;
419     }
420     if(flag==9){
421         clrmos(MouseX,MouseY);
422         barword32(465,400,BLUE,4,4,"轨迹查询",LIGHTGRAY);
423         flag=0;
424     }
425     if(flag==11){

```

```

425         if(flag==113){
426             clrmos(MouseX,MouseY);
427             returnbutton(LIGHTGRAY);
428             flag=0;
429         }
430     }
431     if(MouseS!=0){
432         MouseS=0;
433     }
434
435 }
436
437 }
438
439 void pagepapostrecord(int *page,int *pakey,int *trkey){
440     int flag=0;
441     char trainname[5]={'\0'};
442     int yy;
443     int mm;
444     int dd;
445     int date;
446
447     clrmos(MouseX,MouseY);
448     delay(100);
449     save_bk_mou(MouseX,MouseY);
450
451     drawpapostrecord();
452
453     while(1){
454         newmouse(&MouseX,&MouseY,&press);
455
456         if(inreturnbutton()==1){
457             if(pressreturnbutton()==2){
458                 MouseS=1;
459                 if (flag!=113){
460                     clrmos(MouseX,MouseY);
461                     delay(10);
462                     returnbutton(RED);
463                     flag=113;
464                 }
465                 continue;
466             }
467             else if(pressreturnbutton()==1){
468                 returnbutton(RED);
469                 MouseS=0;
470                 *page=5;
471                 break;
472             }

```

```

473     }
474
475     if(inbarwordframe(280,185,WHITE,3,"",0,RED)){
476         if(pressbarwordframe(280,185,WHITE,3,"",0,RED)==2){
477             MouseS=2;
478             if(flag!=1){
479                 flag=1;
480             }
481             continue;
482         }
483         if(pressbarwordframe(280,185,WHITE,3,"",0,RED)==1){
484             MouseS=0;
485             memset(trainname,'\0',sizeof(trainname));
486             barwordframe(280,185,WHITE,3,"",0,RED);
487             Input_Vis(trainname+1,280,185,3,WHITE,BLACK);
488             continue;
489         }
490     }
491
492     if(inbarword32(430,185,BLUE,2,2,"确认",WHITE)){
493         if(pressbarword32(430,185,BLUE,2,2,"确认",WHITE)==2){
494             MouseS=1;
495             if(flag!=2){
496                 clrmous(MouseX,MouseY);
497                 delay(10);
498                 barword32(430,185,BLUE,2,2,"确认",RED);
499                 flag=2;
500             }
501             continue;
502         }
503         else if(pressbarword32(430,185,BLUE,2,2,"确认",WHITE)==1){
504             MouseS=0;
505             barword32(430,185,BLUE,2,2,"确认",RED);
506             gettodaydate(&yy,&mm,&dd);
507             postdate(yy,mm,dd,&date);
508             trainname[0]='G';
509             if(getpostedtrkey(date,trainname,trkey)!=1){
510                 recordzero();
511                 delay(300);
512                 *page=6;
513                 break;
514             }
515             delay(10);
516             MouseS=0;
517             *page=8;
518             break;;
519         }
520     }

```

```

521
522     if(flag){
523         if(flag==2){
524             clrmos(MouseX,MouseY);
525             barword32(430,185,BLUE,2,2,"确认",WHITE);
526             flag=0;
527         }
528         if(flag==113){
529             clrmos(MouseX,MouseY);
530             returnbutton(LIGHTGRAY);
531             flag=0;
532         }
533     }
534     if(MouseS!=0){
535         MouseS=0;
536     }
537 }
538 }
539
540 void pagepagetrecord(int *page,int *pakey){
541     int flag;
542     int trkeyset[20]={0};
543     struct train *ptrnu;
544     struct train *ptrnm;
545     struct train *ptrnl;
546     int i;
547
548     int currentpage=1;
549     int countlabel=getrecordbypakey(*pakey,trkeyset);
550     int countpage=(countlabel-1)/3+1;
551
552     int u=0,m=0,l=0;
553
554     ptrnu=(struct train *)malloc(sizeof(TRAIN));
555     ptrnm=(struct train *)malloc(sizeof(TRAIN));
556     ptrnl=(struct train *)malloc(sizeof(TRAIN));
557
558     u=0;
559     m=1;
560     l=2;
561     gettrainbytrkey(trkeyset[u],ptrnu);
562     gettrainbytrkey(trkeyset[m],ptrnm);
563     gettrainbytrkey(trkeyset[l],ptrnl);
564
565     if(u>=countlabel){
566         ptrnu=NULL;
567     }

```

```
568     if(m>=countlabel){
569         ptrnm=NULL;
570     }
571     if(l>=countlabel){
572         ptrnl=NULL;
573     }
574
575     clrmous(MouseX,MouseY);
576     delay(100);
577     save_bk_mou(MouseX,MouseY);
578
579     drawpagetrecord(ptrnu,ptrnm,ptrnl,currentpage,countpage);
580
581     while(1){
582         newmouse(&MouseX,&MouseY,&press);
583         if(inreturnbutton()==1){
584             if(pressreturnbutton()==2){
585                 MouseS=1;
586                 if (flag!=113){
587                     clrmous(MouseX,MouseY);
588                     delay(10);
589                     returnbutton(RED);
590                     flag=113;
591                 }
592                 continue;
593             }
594             else if(pressreturnbutton()==1){
595                 returnbutton(RED);
596                 MouseS=0;
597                 *page=5;
598                 free(ptrnu);
599                 free(ptrnm);
600                 free(ptrnl);
601                 break;
602             }
603         }
604         if(inleftarrow()==1&&currentpage!=1){
605             if(pressleftarrow()==2){
606                 MouseS=1;
607                 if (flag!=1){
608                     clrmous(MouseX,MouseY);
609                     delay(10);
610                     leftarrow(RED);
611                     flag=1;
612                 }
613                 continue;
614             }
615             else if(pressleftarrow()==1){
```

```

616         MouseS=0;
617         u-=3;
618         m-=3;
619         l-=3;
620         free(ptrnu);
621         free(ptrnm);
622         free(ptrnl);
623         ptrnu=(struct train *)malloc(sizeof(TRAIN));
624         ptrnm=(struct train *)malloc(sizeof(TRAIN));
625         ptrnl=(struct train *)malloc(sizeof(TRAIN));
626         //question
627         gettrainbytrkey(trkeyset[u],ptrnu);
628         gettrainbytrkey(trkeyset[m],ptrnm);
629         gettrainbytrkey(trkeyset[l],ptrnl);
630         currentpage--;
631         drawpagetrecord(ptrnu,ptrnm,ptrnl,currentpage,countpage);
632         continue;
633     }
634 }
635 if(inrightarrow()==1&&currentpage!=countpage){
636     if(pressrightarrow()==2){
637         MouseS=1;
638         if (flag!=2){
639             clrmous(MouseX,MouseY);
640             delay(10);
641             rightarrow(RED);
642             flag=2;
643         }
644         continue;
645     }
646     else if(pressrightarrow()==1){
647         MouseS=0;
648         u+=3;
649         m+=3;
650         l+=3;
651         free(ptrnu);
652         free(ptrnm);
653         free(ptrnl);
654         ptrnu=(struct train *)malloc(sizeof(TRAIN));
655         ptrnm=(struct train *)malloc(sizeof(TRAIN));
656         ptrnl=(struct train *)malloc(sizeof(TRAIN));
657         gettrainbytrkey(trkeyset[u],ptrnu);
658         gettrainbytrkey(trkeyset[m],ptrnm);
659         gettrainbytrkey(trkeyset[l],ptrnl);
660         if(u>=countlabel){
661             ptrnu=NULL;
662         }
663         if(m>=countlabel){

```



```

663         if (l >= countlabel) {
664             ptrnm=NULL;
665         }
666         if (l >= countlabel) {
667             ptrnl=NULL;
668         }
669         currentpage++;
670         drawpagetrecord(ptrnu,ptrnm,ptrnl,currentpage,countpage);
671         continue;
672     }
673 }
674 if(flag){
675     if(flag==113){
676         clrmous(MouseX,MouseY);
677         returnbutton(LIGHTGRAY);
678         flag=0;
679     }
680     if(flag==1){
681         clrmous(MouseX,MouseY);
682         leftarrow(BLUE);
683         flag=0;
684     }
685     if(flag==2){
686         clrmous(MouseX,MouseY);
687         rightarrow(BLUE);
688         flag=0;
689     }
690 }
691 if(MouseS!=0){
692     MouseS=0;
693 }
694 }
695 }
696
697 void pageposttrack(int *page,int *pakey,int *trkey){
698     int flag=0;
699     char track[6];
700     struct train trn;
701
702     gettrainbytrkey(*trkey,&trn);
703
704     clrmous(MouseX,MouseY);
705     delay(100);
706     save_bk_mou(MouseX,MouseY);
707
708     drawstationselect(trn.trainname);
709     while(1){
710         newmouse(&MouseX,&MouseY,&press);

```

```

711         if(inreturnbutton()==1){
712             if(pressreturnbutton()==2){
713                 MouseS=1;
714                 if (flag!=113){
715                     clrmos(MouseX,MouseY);
716                     delay(10);
717                     returnbutton(RED);
718                     flag=113;
719                 }
720                 continue;
721             }
722             else if(pressreturnbutton()==1){
723                 returnbutton(RED);
724                 MouseS=0;
725                 *page=6;
726                 break;
727             }
728         }
729         if(intrack1()){
730             if(presstrack1()==2){
731                 MouseS=1;
732                 if(flag!=1){
733                     clrmos(MouseX,MouseY);
734                     delay(10);
735                     track1(trn.trainname,RED);
736                     flag=1;
737                 }
738                 continue;
739             }
740             if(presstrack1()==1){
741                 MouseS=0;
742                 track1(trn.trainname,RED);
743                 gettrackstring(trn.trainname,1,track);
744                 if(writerecordv1(*pakey,*trkey,track)==-1){
745                     trackrecordrepeat();
746                     delay(300);
747                     *page=8;
748                     break;
749                 }
750                 else{
751                     trackrecordok();
752                     delay(300);
753                     *page=5;
754                     break;
755                 }
756             }
757         }
758         if(intrack2()){

```

```

759         if(presstrack2()==2){
760             MouseS=1;
761             if(flag!=2){
762                 clrmos(MouseX,MouseY);
763                 delay(10);
764                 track2(trn.trainname,RED);
765                 flag=2;
766             }
767             continue;
768         }
769         if(presstrack2()==1){
770             MouseS=0;
771             track2(trn.trainname,RED);
772             gettrackstring(trn.trainname,2,track);
773             if(writerecordv1(*pakey,*trkey,track)==-1){
774                 trackrecordrepeat();
775                 delay(300);
776                 *page=8;
777                 break;
778             }
779             else{
780                 trackrecordok();
781                 delay(300);
782                 *page=5;
783                 break;
784             }
785         }
786     }
787     if(intrack3()){
788         if(presstrack3()==2){
789             MouseS=1;
790             if(flag!=3){
791                 clrmos(MouseX,MouseY);
792                 delay(10);
793                 track3(trn.trainname,RED);
794                 flag=3;
795             }
796             continue;
797         }
798         if(presstrack3()==1){
799             MouseS=0;
800             track3(trn.trainname,RED);
801             gettrackstring(trn.trainname,3,track);
802             if(writerecordv1(*pakey,*trkey,track)==-1){
803                 trackrecordrepeat();
804                 delay(300);
805                 *page=8;
806                 break;

```

```

806         break;
807     }
808     else{
809         trackrecordok();
810         delay(300);
811         *page=5;
812         break;
813     }
814 }
815 }
816 if(intrack4()){
817     if(presstrack4()==2){
818         MouseS=1;
819         if(flag!=4){
820             clrmos(MouseX,MouseY);
821             delay(10);
822             track4(trn.trainname,RED);
823             flag=4;
824         }
825         continue;
826     }
827     if(presstrack4()==1){
828         MouseS=0;
829         track4(trn.trainname,RED);
830         gettrackstring(trn.trainname,4,track);
831         if(writerecordv1(*pakey,*trkey,track)==-1){
832             trackrecordrepeat();
833             delay(300);
834             *page=8;
835             break;
836         }
837         else{
838             trackrecordok();
839             delay(300);
840             *page=5;
841             break;
842         }
843     }
844 }
845 if(intrack5()){
846     if(presstrack5()==2){
847         MouseS=1;
848         if(flag!=5){
849             clrmos(MouseX,MouseY);
850             delay(10);
851             track5(trn.trainname,RED);
852             flag=5;
853         }

```

```

854         continue;
855     }
856     if(presstrack5()==1){
857         MouseS=0;
858         track5(trn.trainname,RED);
859         gettrackstring(trn.trainname,5,track);
860         if(writerecordv1(*pakey,*trkey,track)==-1){
861             trackrecordrepeat();
862             delay(300);
863             *page=8;
864             break;
865         }
866         else{
867             trackrecordok();
868             delay(300);
869             *page=5;
870             break;
871         }
872     }
873 }
874 if(intrack6()){
875     if(presstrack6()==2){
876         MouseS=1;
877         if(flag!=6){
878             clrmous(MouseX,MouseY);
879             delay(10);
880             track6(trn.trainname,RED);
881             flag=6;
882         }
883         continue;
884     }
885     if(presstrack6()==1){
886         MouseS=0;
887         track6(trn.trainname,RED);
888         gettrackstring(trn.trainname,6,track);
889         if(writerecordv1(*pakey,*trkey,track)==-1){
890             trackrecordrepeat();
891             delay(300);
892             *page=8;
893             break;
894         }
895         else{
896             trackrecordok();
897             delay(300);
898             *page=5;
899             break;
900         }
901     }

```

```

902     }
903     if(intrack7()){
904         if(presstrack7()==2){
905             MouseS=1;
906             if(flag!=7){
907                 clrmous(MouseX,MouseY);
908                 delay(10);
909                 track7(trn.trainname,RED);
910                 flag=7;
911             }
912             continue;
913         }
914         if(presstrack7()==1){
915             MouseS=0;
916             track7(trn.trainname,RED);
917             gettrackstring(trn.trainname,7,track);
918             if(writerecordv1(*pakey,*trkey,track)==-1){
919                 trackrecordrepeat();
920                 delay(300);
921                 *page=8;
922                 break;
923             }
924             else{
925                 trackrecordok();
926                 delay(300);
927                 *page=5;
928                 break;
929             }
930         }
931     }
932     if(intrack8()){
933         if(presstrack8()==2){
934             MouseS=1;
935             if(flag!=8){
936                 clrmous(MouseX,MouseY);
937                 delay(10);
938                 track8(trn.trainname,RED);
939                 flag=8;
940             }
941             continue;
942         }
943         if(presstrack8()==1){
944             MouseS=0;
945             track8(trn.trainname,RED);
946             gettrackstring(trn.trainname,8,track);
947             if(writerecordv1(*pakey,*trkey,track)==-1){
948                 trackrecordrepeat();
949                 delay(300);

```

```

949         delay(300);
950         *page=8;
951         break;
952     }
953     else{
954         trackrecordok();
955         delay(300);
956         *page=5;
957         break;
958     }
959 }
960 }
961 if(intrack9()){
962     if(presstrack9()==2){
963         MouseS=1;
964         if(flag!=9){
965             clrmos(MouseX,MouseY);
966             delay(10);
967             track9(trn.trainname,RED);
968             flag=9;
969         }
970         continue;
971     }
972     if(presstrack9()==1){
973         MouseS=0;
974         track9(trn.trainname,RED);
975         gettrackstring(trn.trainname,9,track);
976         if(writerecordv1(*pakey,*trkey,track)==-1){
977             trackrecordrepeat();
978             delay(300);
979             *page=8;
980             break;
981         }
982         else{
983             trackrecordok();
984             delay(300);
985             *page=5;
986             break;
987         }
988     }
989 }
990 if(intrack10()){
991     if(presstrack10()==2){
992         MouseS=1;
993         if(flag!=10){
994             clrmos(MouseX,MouseY);
995             delay(10);
996             track10(trn.trainname,RED);

```

```
997         flag=10;
998     }
999     continue;
1000 }
1001 if(presstrack10()==1){
1002     MouseS=0;
1003     track10(trn.trainname,RED);
1004     gettrackstring(trn.trainname,10,track);
1005     if(writerecordv1(*pakey,*trkey,track)==-1){
1006         trackrecordrepeat();
1007         delay(300);
1008         *page=8;
1009         break;
1010     }
1011     else{
1012         trackrecordok();
1013         delay(300);
1014         *page=5;
1015         break;
1016     }
1017 }
1018 }
1019
1020 if(flag){
1021     if(flag==1){
1022         clrmous(MouseX,MouseY);
1023         track1(trn.trainname,WHITE);
1024         flag=0;
1025     }
1026     if(flag==113){
1027         clrmous(MouseX,MouseY);
1028         returnbutton(LIGHTGRAY);
1029         flag=0;
1030     }
1031     if(flag==2){
1032         clrmous(MouseX,MouseY);
1033         track2(trn.trainname,WHITE);
1034         flag=0;
1035     }
1036     if(flag==3){
1037         clrmous(MouseX,MouseY);
1038         track3(trn.trainname,WHITE);
1039         flag=0;
1040     }
1041     if(flag==4){
1042         clrmous(MouseX,MouseY);
1043         track4(trn.trainname,WHITE);
1044         flag=0;
```



```

1045     }
1046     if(flag==5){
1047         clrmos(MouseX,MouseY);
1048         track5(trn.trainname,WHITE);
1049         flag=0;
1050     }
1051     if(flag==6){
1052         clrmos(MouseX,MouseY);
1053         track6(trn.trainname,WHITE);
1054         flag=0;
1055     }
1056     if(flag==7){
1057         clrmos(MouseX,MouseY);
1058         track7(trn.trainname,WHITE);
1059         flag=0;
1060     }
1061     if(flag==8){
1062         clrmos(MouseX,MouseY);
1063         track8(trn.trainname,WHITE);
1064         flag=0;
1065     }
1066     if(flag==9){
1067         clrmos(MouseX,MouseY);
1068         track9(trn.trainname,WHITE);
1069         flag=0;
1070     }
1071     if(flag==10){
1072         clrmos(MouseX,MouseY);
1073         track10(trn.trainname,WHITE);
1074         flag=0;
1075     }
1076 }
1077 if(MouseS!=0){
1078     MouseS=0;
1079 }
1080 }
1081
1082 }
1083
1084 void pagepagetrack(int *page,int *pakey){
1085     int flag;
1086     char trackset[10][10];
1087     int i;
1088     int trackcount=gettrackbypakey(*pakey,trackset);
1089
1090     clrmos(MouseX,MouseY);
1091     delay(100);

```

```

1092     save_bk_mou(MouseX,MouseY);
1093
1094     drawpagetrack();
1095
1096     for(i=0;i<trackcount;i++){
1097         mapline(trackset[i],RED);
1098     }
1099
1100     while(1){
1101         newmouse(&MouseX,&MouseY,&press);
1102         if(inreturnbutton()==1){
1103             if(pressreturnbutton()==2){
1104                 MouseS=1;
1105                 if (flag!=113){
1106                     clrmous(MouseX,MouseY);
1107                     delay(10);
1108                     returnbutton(RED);
1109                     flag=113;
1110                 }
1111                 continue;
1112             }
1113             else if(pressreturnbutton()==1){
1114                 returnbutton(RED);
1115                 MouseS=0;
1116                 *page=5;
1117                 break;
1118             }
1119         }
1120         if(flag){
1121             if(flag==113){
1122                 clrmous(MouseX,MouseY);
1123                 returnbutton(LIGHTGRAY);
1124                 flag=0;
1125             }
1126         }
1127         if(MouseS!=0){
1128             MouseS=0;
1129         }
1130     }
1131 }

```

十二、时间安排与分工

表格视图 1
 ▼

<div> <div>🔒</div> <div>A≡ 多行文本</div> </div>	A≡ 多行文本 1	⌵ 单选
1		
2		
3		

3 条记录