

**课 程 设 计 报 告**

**题目：个人自助游管理信息系统**

**课程名称： C语言程序设计**

**专业班级： IOT1601**

**学 号： U201614898**

**姓 名： 潘翔**

**指导教师： 刘芳**

**报告日期： 2017/9/12**

**计算机科学与技术学院**

目录

[一．系统功能模块结构 4](#_Toc493858630)

[二．数据结构设计及用法说明 5](#_Toc493858631)

[3.1行程数据结构(Route) 5](#_Toc493858632)

[3.2每日信息数据结构(DayInfo) 5](#_Toc493858633)

[3.3每日消费信息数据结构(DayEXPInfo) 6](#_Toc493858634)

[3.5用法 6](#_Toc493858635)

[3.6数据类型及示例 6](#_Toc493858636)

[3.6.1行程： 6](#_Toc493858637)

[3.6.2每日信息： 7](#_Toc493858638)

[3.6.3每日消费信息： 7](#_Toc493858639)

[3.6.4三级链表结构 9](#_Toc493858640)

[三．程序结构(流程图) 10](#_Toc493858641)

[3.1文件模块运行流程图 10](#_Toc493858642)

[3.1.1文件读取操作流程图 10](#_Toc493858643)

[3.1.2文件保存操作流程图 11](#_Toc493858644)

[3.2插入模块运行流程图 12](#_Toc493858645)

[3.2.1插入日程流程图 12](#_Toc493858646)

[3.2.2插入每日信息流程图 13](#_Toc493858647)

[3.2.3插入每日消费信息流程图 14](#_Toc493858648)

[3.3修改模块运行流程图 15](#_Toc493858649)

[3.4删除模块运行流程图 16](#_Toc493858650)

[3.4.1删除日程流程图 16](#_Toc493858651)

[3.4.2删除每日信息流程图 17](#_Toc493858652)

[3.4.3删除每日消费信息流程图 18](#_Toc493858653)

[3.5显示模块运行流程图 20](#_Toc493858654)

[3.6查询模块运行流程图 20](#_Toc493858655)

[3.6.1查询行程流程图 20](#_Toc493858656)

[3.6.2查询每日信息流程图 21](#_Toc493858657)

[3.6.3查询每日交易信息流程图 22](#_Toc493858658)

[3.7显示模块运行流程图 23](#_Toc493858659)

[3.8统计模块运行流程图 23](#_Toc493858660)

[四．各模块的功能 25](#_Toc493858661)

[4.1文件模块功能 25](#_Toc493858662)

[4.2插入模块功能 25](#_Toc493858663)

[4.3修改模块功能 25](#_Toc493858664)

[4.4删除模块功能 25](#_Toc493858665)

[4.5统计模块功能 25](#_Toc493858666)

[4.6查询模块功能 25](#_Toc493858667)

[4.7显示模块功能 25](#_Toc493858668)

[五．程序测试 26](#_Toc493858669)

[5.1文件模块测试 26](#_Toc493858670)

[5.2插入模块测试 26](#_Toc493858671)

[5.3修改模块测试 26](#_Toc493858672)

[5.4删除模块测试 26](#_Toc493858673)

[5.5统计模块测试 26](#_Toc493858674)

[5.6查询模块测试 26](#_Toc493858675)

[5.7显示模块测试 26](#_Toc493858676)

[六．实验体会 26](#_Toc493858677)

[七．鸣谢 27](#_Toc493858678)

[八．参考文献 27](#_Toc493858679)

[附录：源代码清单 27](#_Toc493858680)

[CourseDesign.pro 27](#_Toc493858681)

[Personaltrip.h 28](#_Toc493858682)

[PersonalTrip.cpp 31](#_Toc493858683)

[newRoute.h 58](#_Toc493858684)

[newRoute.cpp 59](#_Toc493858685)

[newDayInfo.h 60](#_Toc493858686)

[newDayInfo.cpp 61](#_Toc493858687)

[newDayEXPInfo.h 62](#_Toc493858688)

[newDayEXPInfo.cpp 63](#_Toc493858689)

[mainwindow.h 64](#_Toc493858690)

[mainwindow.cpp 67](#_Toc493858691)

[main.cpp 104](#_Toc493858692)

## 一．系统功能模块结构

个人自助游管理信息系统由七大功能模块组成：文件模块，插入模块，修改模块，删除模块，统计模块，查询模块，显示模块。如下图所示：

## 二．数据结构设计及用法说明

### 3.1行程数据结构(Route)

typedef struct Route //行程一级链表结构体

{

int routeNum;//行程编号

//char routeNum[10];//行程编号

char routeName[20];//主要目的地名称

char routeCountry[20];//目的地所属国家/省份

DressingIndex dressingIndex;

char bigTrans[20];//大交通 主要交通

char mainScene[200];//主要景点

char remoteNum[10];//异地办理手续行程编号 此处更改为char

//如需异地面签或提交材料，给出异地办手续的行程编号，否则为空

char beginDate[20];//出发日期

char arriveDate[20];//返回到达日期

int totalTime;//行程总天数

int totalNum;//行程总人数

float totalPrice;//行程总费用

struct DayInfo\* hDayInfo;//行程具体信息头指针

struct Route\* nextRoute;//链表next指针

}Route, \*p\_Route;

### 3.2每日信息数据结构(DayInfo)

typedef struct DayInfo //行程具体信息结构体

{

int routeNum;//所属行程编号

//char routeNum[10];//所属行程编号

int order;//序号 2代表第二天 存成int

char breakfastLoc[50];//早餐地点

char morningScene[50];//上午景点

char LunchLoc[50];//午餐地点

char afternoonScene[50];//下午景点

char dinnerLoc[50];//晚餐地点

char transInfo[200];//小交通

char residence[50];//住宿地点

struct DayEXPInfo\* hDayEXPInfo;

struct DayInfo\* nextDayInfo;//链表next指针

}DayInfo, \*p\_DayInfo;

### 3.3每日消费信息数据结构(DayEXPInfo)

typedef struct DayEXPInfo //每天消费信息

{

int routeNum;

//char routeNum[10];//所属行程编号

int order;//序号 2代表第二天

int serialNum;//顺序号

char TxTime[20];//交易时间

char TxKind[16];//交易种类

float TxAmount;//交易金额

char isPrepayment[10];//是否预支付

char payment[10];//支付方式

char explain[50];//说明

struct DayEXPInfo\* nextDayEXPInfo;

}DayEXPInfo, \*p\_DayEXPInf

### 3.5用法

以上的结构体均可直接通过malloc函数开辟内存空间，其中char[]类型需使用strcmp赋值，若QT中为Qstring或者String类型需转换后使用strcmp赋值，int,float等亦需要转换后赋值。

### 3.6数据类型及示例

#### 3.6.1行程：

|  |  |  |
| --- | --- | --- |
| **中文字段名** | **类型及长度** | **举例或说明** |
| 行程编号 | int | 顺序号 |
| 主要目的地名称 | char[20] | 喀什 |
| 目的地所属国家/省份 | char[20] | 新疆 |
| 出行期温度及穿衣指数 | struct | 最高-最低温度，着装推荐 |
| 大交通 | char[20] | 火车/飞机/轮船 |
| 主要景点 | char[200] | 喀什高台民居、香妃墓、卡拉库里湖、红旗拉普…… |
| 异地办理手续行程编号 | char[10] | 如需异地面签或提交材料，给出异地办手续的行程编号，否则为空 |
| 出发日期 | char[20] | 2017-9-10 |
| 返回到达日期 | char[20] | 2017-9-20 |
| 行程总天数 | int | 10 |
| 行程总人数 | int | 10 |
| 行程总费用 | float | 9000 |
| 行程具体信息 | Link |  |

#### 3.6.2每日信息：

|  |  |  |
| --- | --- | --- |
| **中文字段名** | **类型及长度** | **举例** |
| 所属行程编号 | int |  |
| 序号 | int | 如2：代表第2天 |
| 早餐地点 | char[50] | 色满宾馆 |
| 上午景点 | char[50] | 喀什高台民居 |
| 午餐地点 | char[50] | 高台民居对面的馕坑烤肉店 |
| 下午景点 | char[50] | 喀什艾提尕尔清真寺 |
| 晚餐地点 | char[50] | 喀什夜市 |
| 小交通路线 | char[200] | 从出发地搭乘公汽/出租/租车等到达目的地的文字说明，可描述一天内的多条路线 |
| 住宿地点 | char[50] | 色满宾馆 |
| 费用明细 | Link |  |

#### 3.6.3每日消费信息：

|  |  |  |
| --- | --- | --- |
| **中文字段名** | **类型及长度** | **举例** |
| 所属行程编号 | int |  |
| 序号 | int | 如2：代表第2天 |
| 明细交易编号 | int | 顺序号 |
| 交易时间 | char[4] | 9:00am: 0900 |
| 交易类别 | char[8] | 餐饮/小交通/大交通/门票/购物/其它 |
| 交易金额 | Float |  |
| 是否预支付 | Char[10] | 是/否 |
| 支付方式 | Char[10] | 现金/卡/支票 |
| 说明 | Char[50] | 如：高台民居4人门票 |

#### 3.6.4三级链表结构

**…**

行程1

head

行程2

行程n

NULL

**…**

Day 1基本信息

Day 2基本信息

Day n基本信息

Day 1的费用明细信息链

Day 2的费用明细信息链

Day 3的费用明细信息链

为无头结点链表

## 三．程序结构(流程图)

### 3.1文件模块运行流程图

#### 3.1.1文件读取操作流程图



#### 3.1.2文件保存操作流程图



### 3.2插入模块运行流程图

#### 3.2.1插入日程流程图



#### 3.2.2插入每日信息流程图



#### 3.2.3插入每日消费信息流程图



### 3.3修改模块运行流程图



### 3.4删除模块运行流程图

#### 3.4.1删除日程流程图



#### 3.4.2删除每日信息流程图



#### 3.4.3删除每日消费信息流程图



### 3.5显示模块运行流程图

### 3.6查询模块运行流程图

#### 3.6.1查询行程流程图



#### 3.6.2查询每日信息流程图



#### 3.6.3查询每日交易信息流程图



### 3.7显示模块运行流程图



### 3.8统计模块运行流程图



## 四．各模块的功能

### 4.1文件模块功能

保存当前的文件，读取文件。

### 4.2插入模块功能

插入三级节点，同时更新树形结构，更新一级节点有关统计信息。

### 4.3修改模块功能

修改三级节点，同时更新树形结构，更新一级节点有关统计信息。

对三级链表进行排序。

### 4.4删除模块功能

删除三级节点，同时更新树形结构，更新一级节点有关统计信息。

### 4.5统计模块功能

统计三级信息，输出统计图，输出费用类型分类统计表格，门票价格最高的行程，日消费最高的行程。

### 4.6查询模块功能

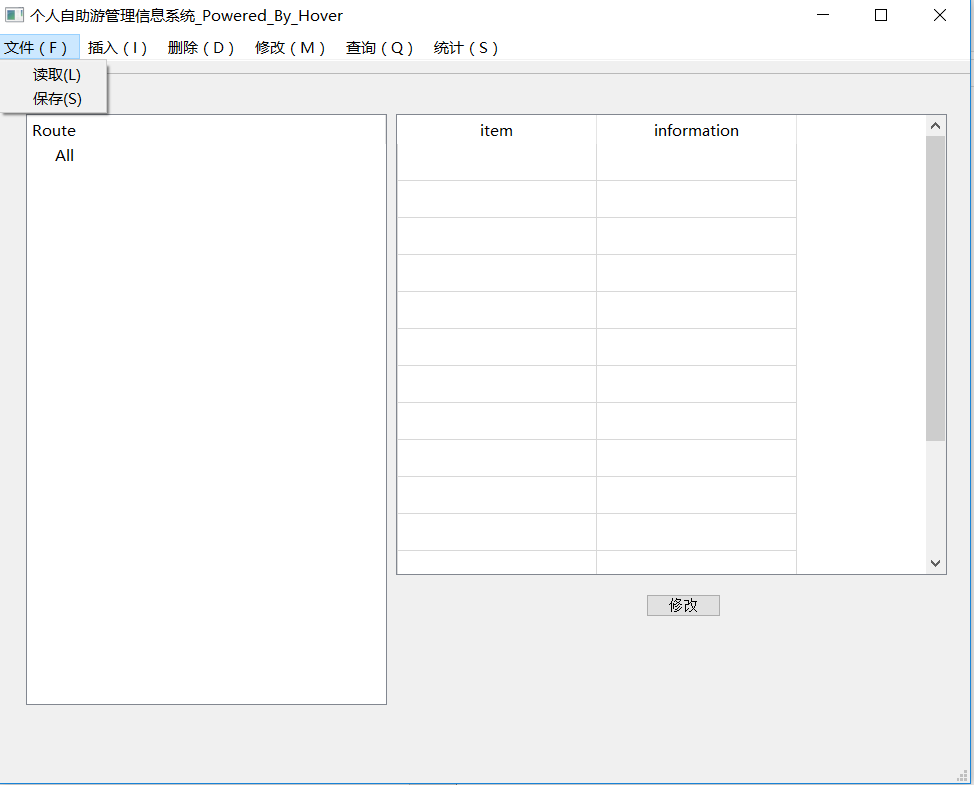
通过编号查询三级链表，查询小交通信息（给定景点），查询穿衣指数（给定时间区间），查询住宿信息（给定旅社名称）。

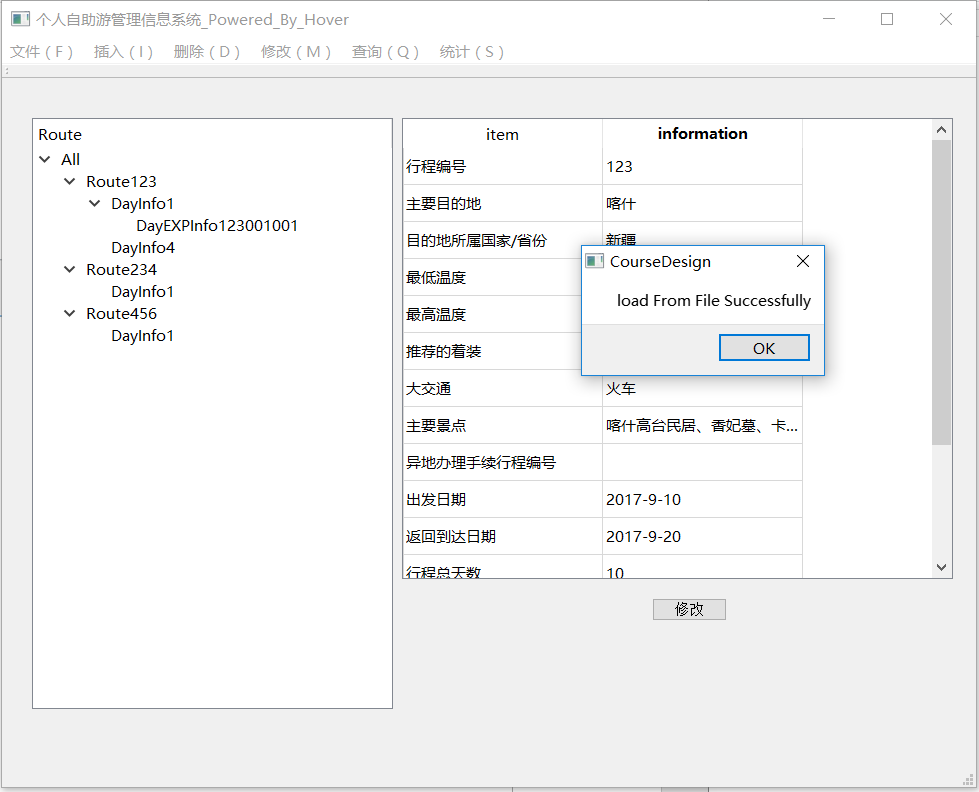
### 4.7显示模块功能

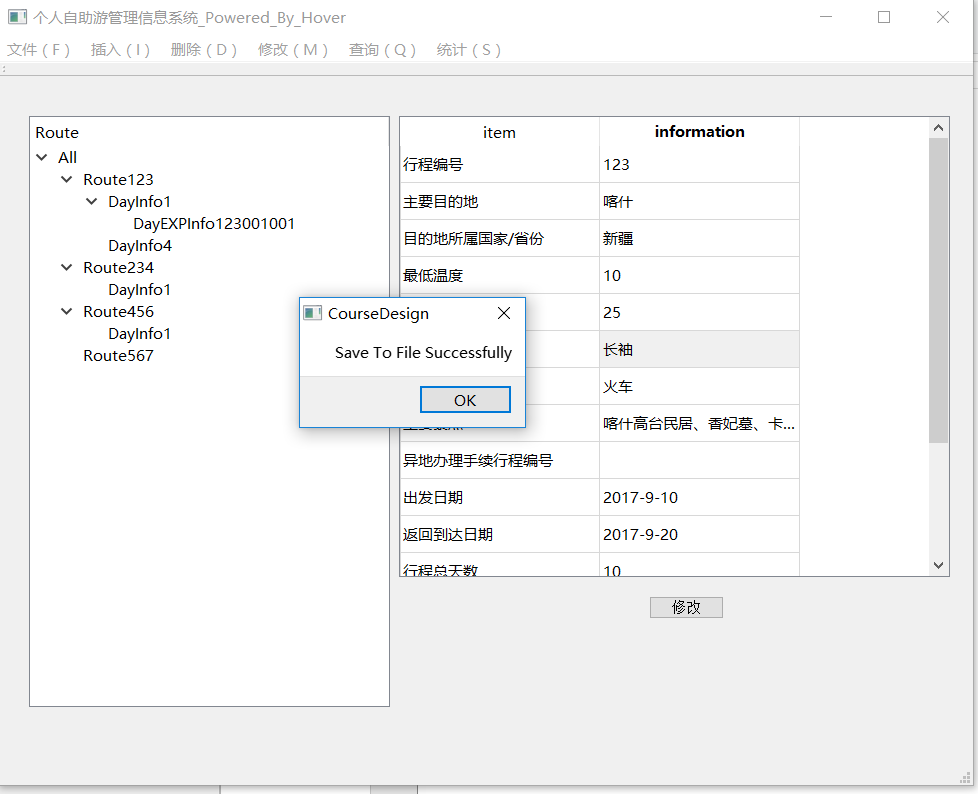
获取鼠标位置，构建树形结构并实时更新，同时配合修改模块。

## 五．程序测试

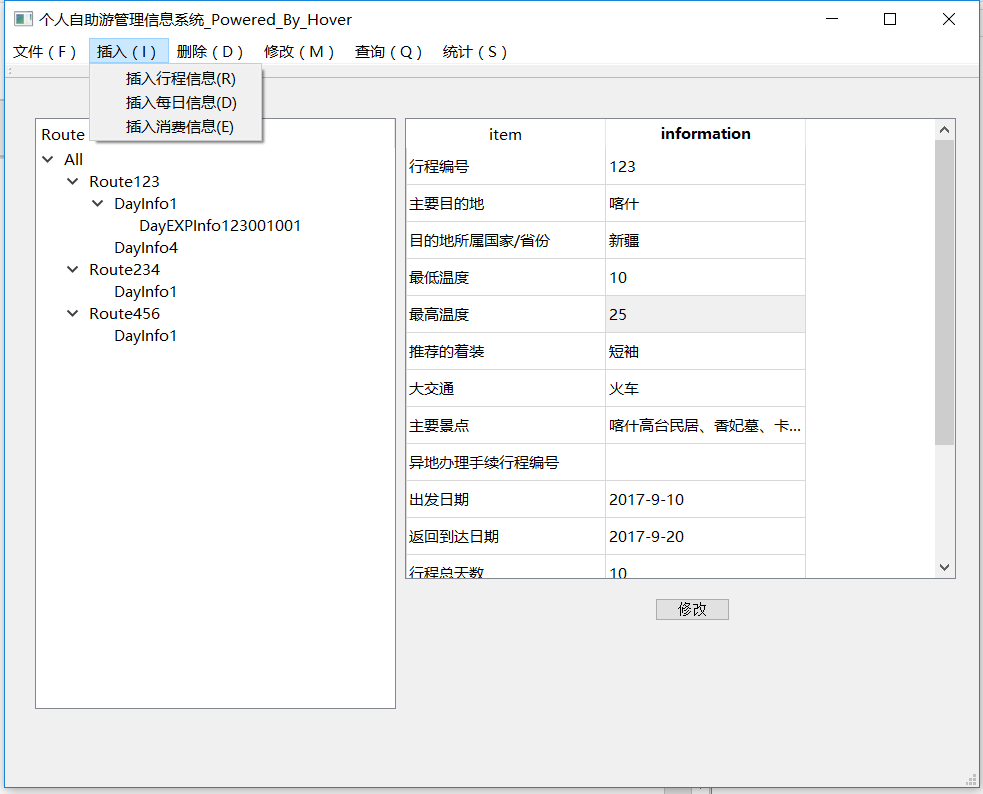
### 5.1文件模块测试

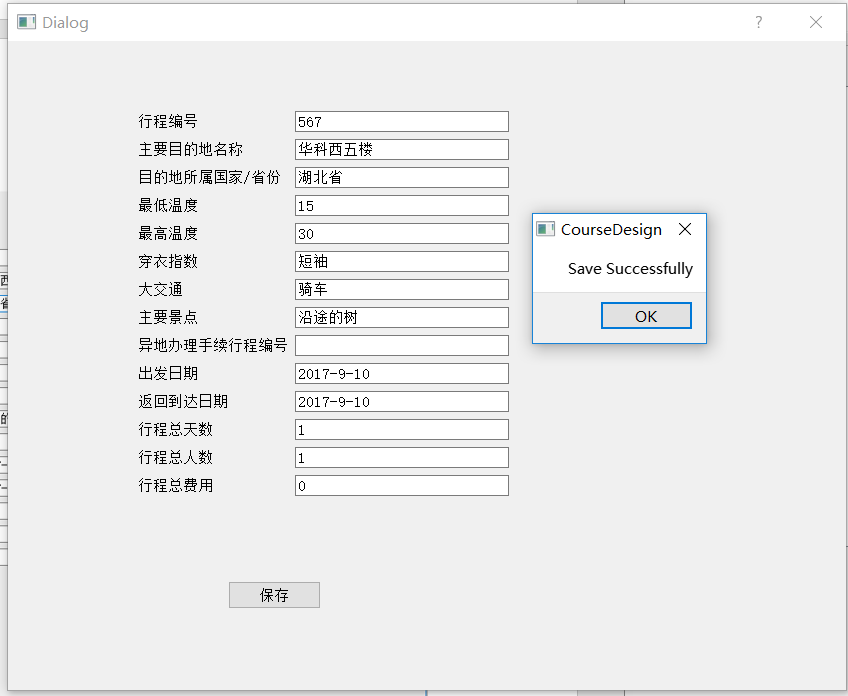


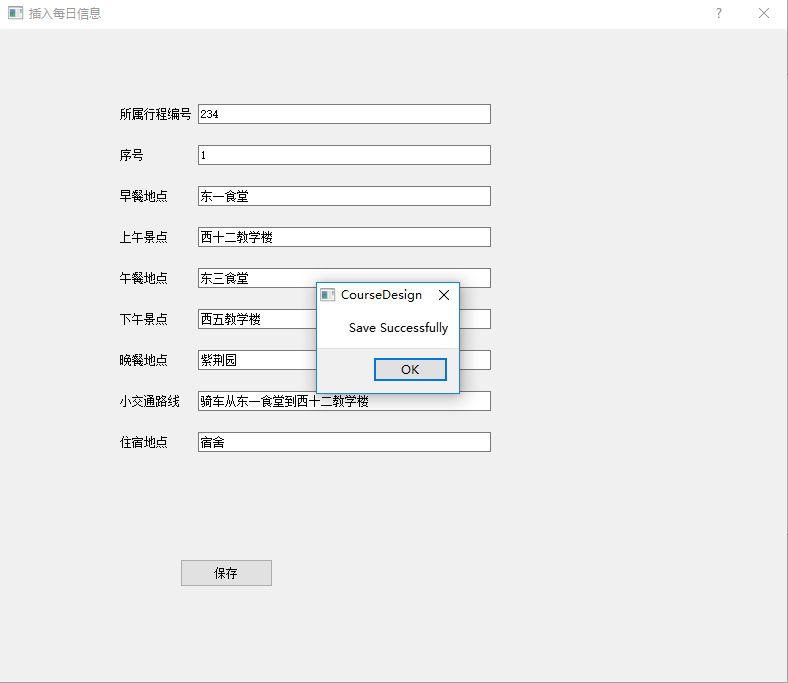


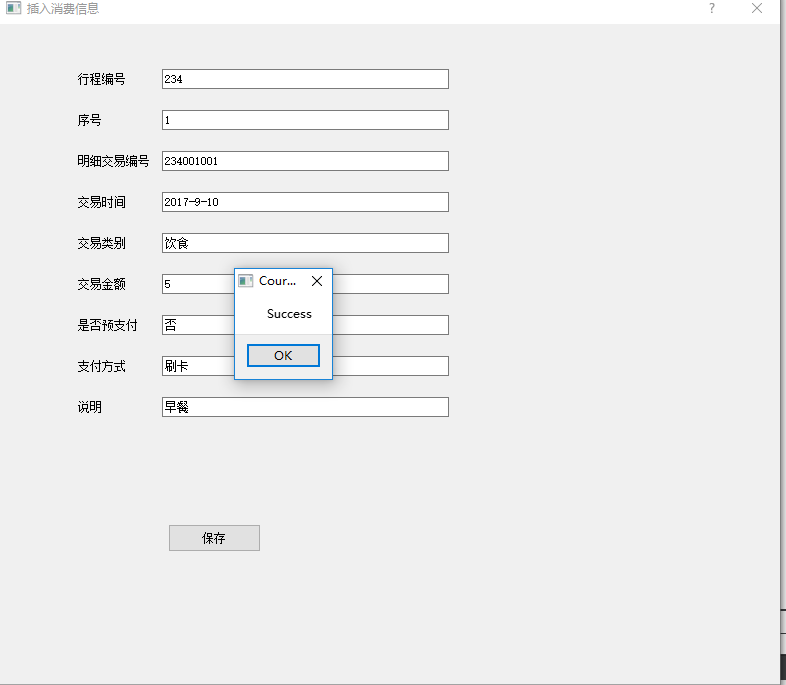


### 5.2插入模块测试

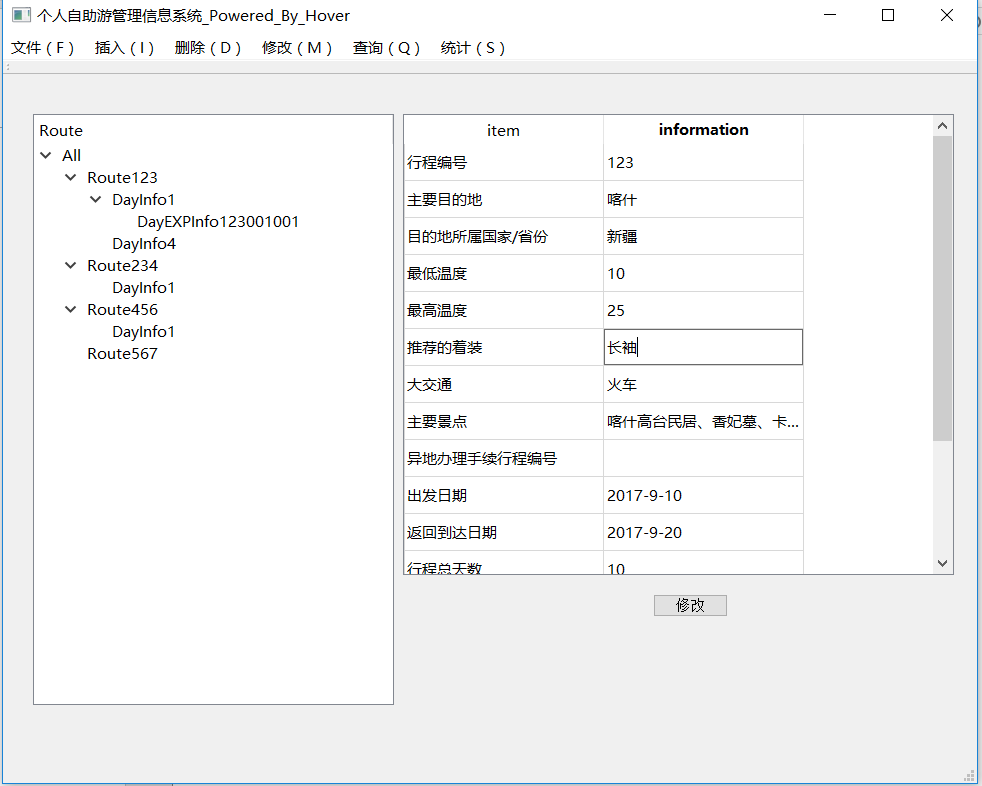








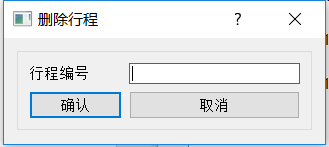
### 5.3修改模块测试

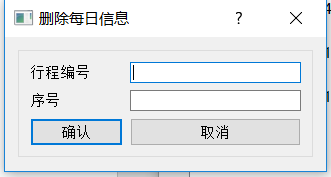


点击修改后

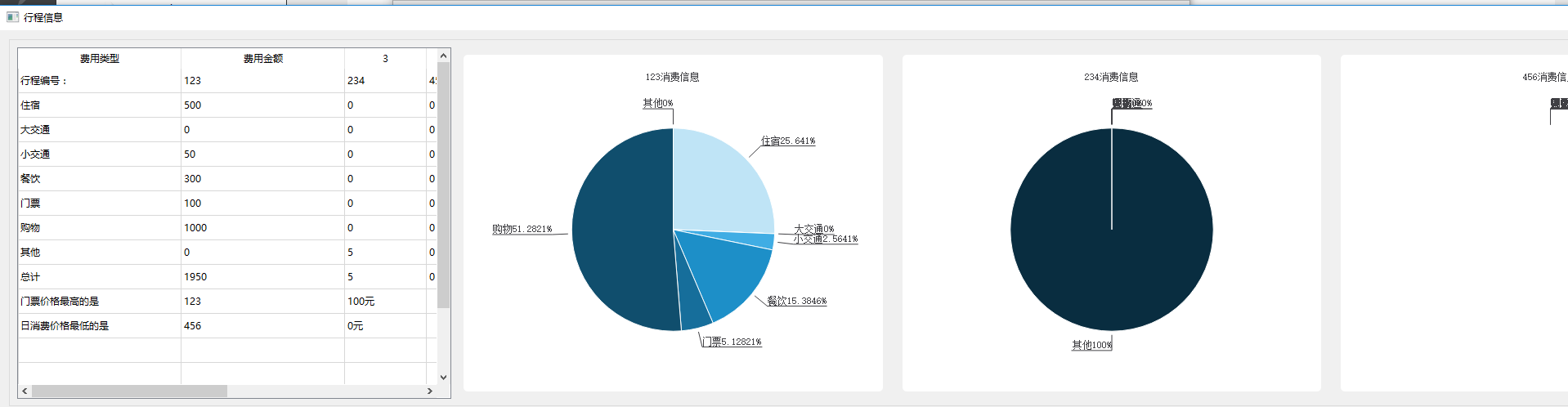


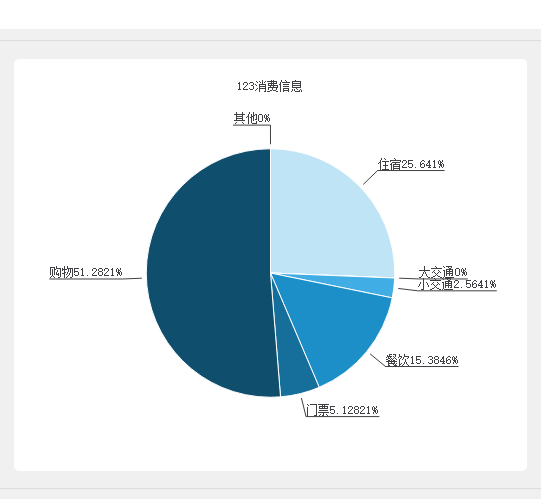
### 5.4删除模块测试





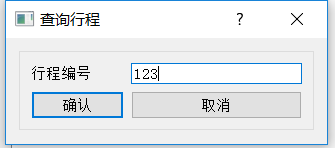
### 5.5统计模块测试

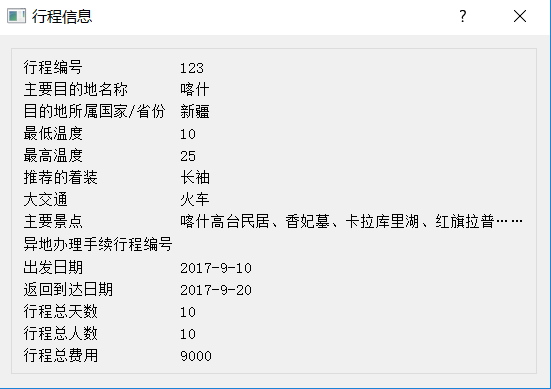


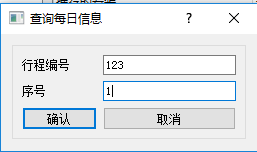


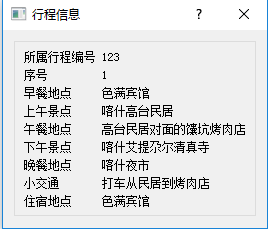


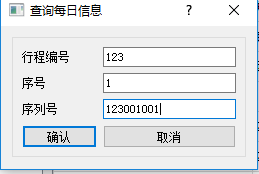
### 5.6查询模块测试







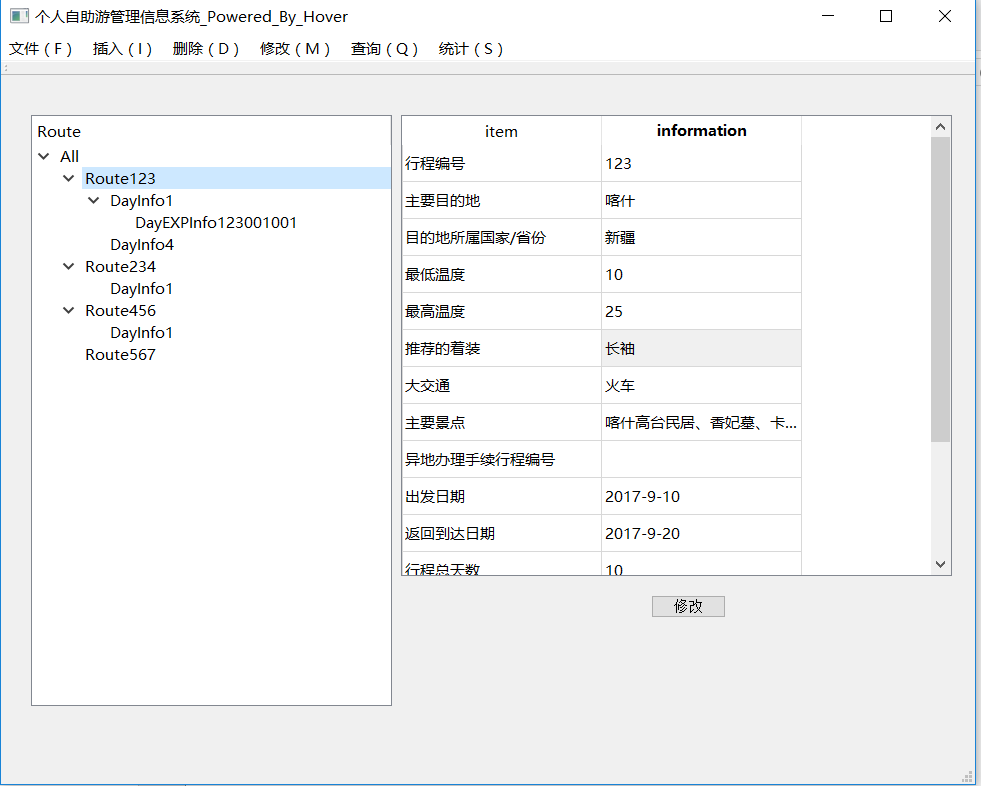


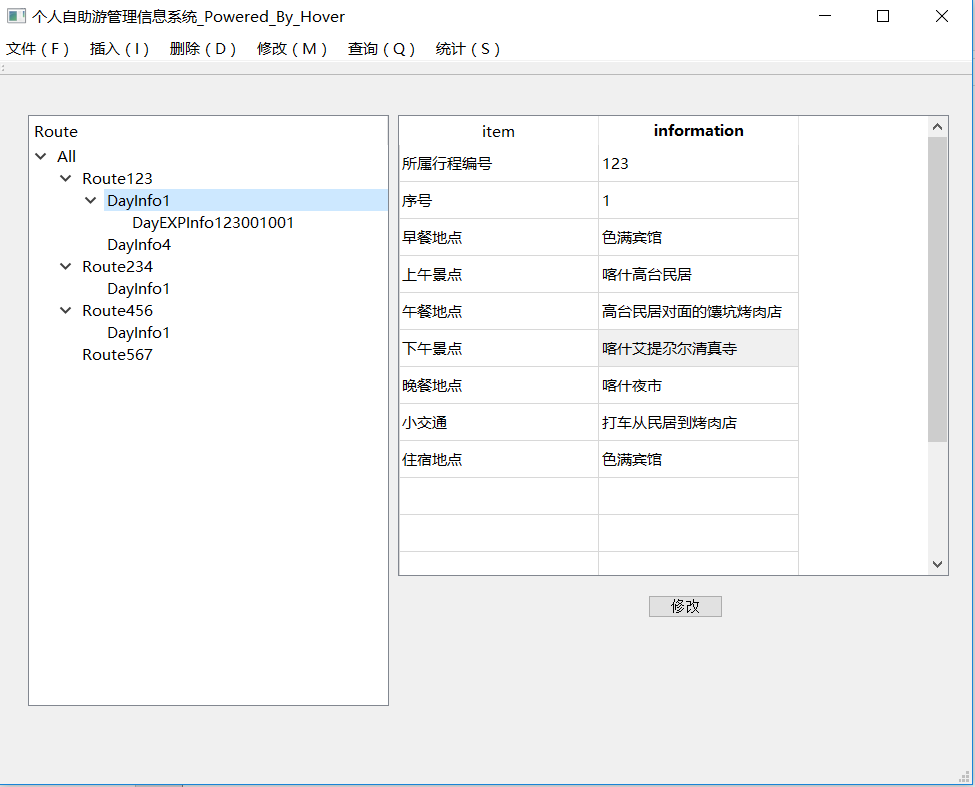


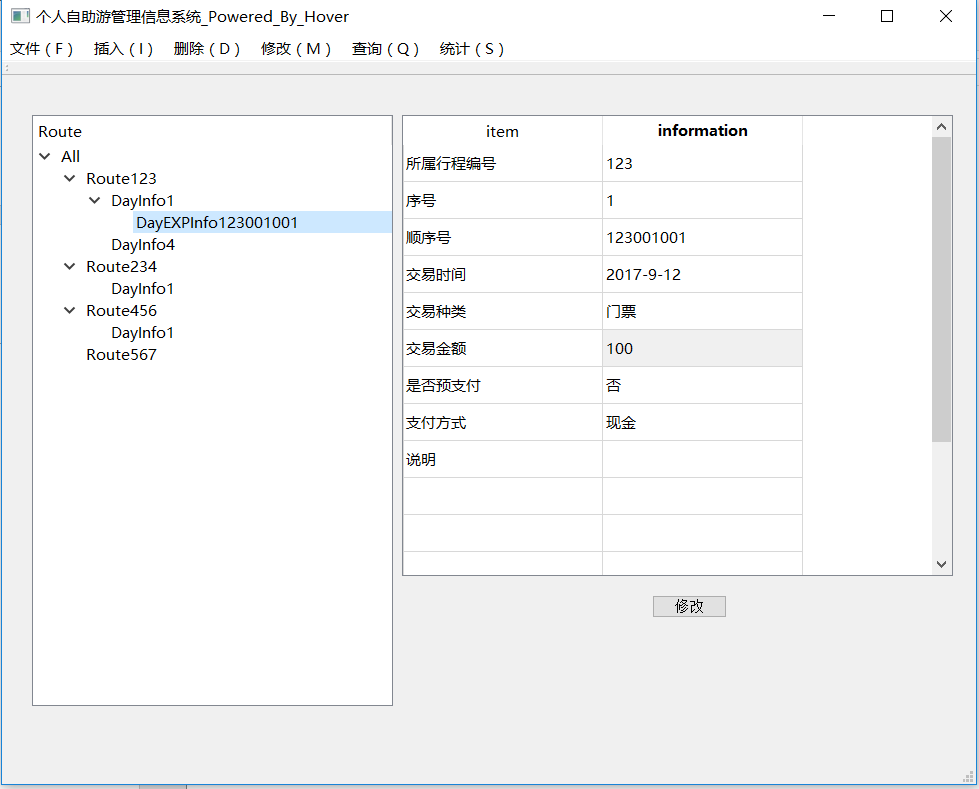


### 5.7显示模块测试

点击左侧不同的item显右侧显示不同的项







## 六．实验体会

在本次课程设计中，使用了C语言实现的程序后台操作和QT实现的界面程序，在核心代码的编写上，并没有花费过多的时间，其中较令人头疼的是C语言不支持重载，若要提供不同的接口要么利用函数指针，要么传入调用的模块参数，但同时需要传入所有的形参，在某些情况下不需要形参的传入造册了一定的空间浪费和编码的不舒服，故只能在界面中进行封装重载进行接口的完善。

在编译器的选择上：开始使用MSVC2017编译器，但由于其是win平台上，故对于UTF-8 无BOM需采用VSCODE或NOTEPAD打开添加BOM，而部分情况下存在中文乱码，早起的更改编码API失效，故采用GCC编译器和unicode编码。

在存储形式的转换上：需要在Qstring String char\*至今进行转换，因为接口的限制，而转换过程中会出现一些问题，故各项的初始化要做好。

在界面的选择上：存在win32API,MFC,GTK,QT等提供选择，由于win32API较为老旧和GTK封装的不良好，而MFC的跨平台特性和应用广度不如QT，故采用QT进行界面设计。

在界面的设计上：采用窗口显示可视化操作和文本查询相结合的方式，使可以采用多种方式增删改查和整体链表结构的可视化，其中可视化部分使用TreeWidget类进行实现，利用链表的遍历构建顺序建立树。

在模块的设计上：本身想要采用opengl嵌入QT窗口进行绘制统计图，但看到QT本身提供绘图模块遂尝试，但由于非自身实现和接口的不熟悉，导致费时更多。其余各模块设计的同时尽量解耦合，在多个模块要调用的函数尽量单独封装。

在调试的过程中，分模块调试清楚之后再进行组装，确保每一模块内部的正确，例如核心代码的非GUI调试。

其中由于C语言的面向过程特性导致编码过程中存在一定的冗余，代码的简洁性不高。

## 七．鸣谢

在本次C语言课程设计构思创作过程中，得到了甘早斌老师和刘芳老师的帮助和支持，在此表示感谢，感谢提供了帮助和支持的老师们！

## 八．参考文献

1.QT官方文档

2.李开, 卢萍, 曹计昌. C语言实验与课程设计, 北京：科学出版社, 2011。

3.CSDN [基于Qt的图表库](http://blog.csdn.net/liang19890820/article/details/54600307)http://blog.csdn.net/liang19890820/article/details/54600307

## 附录：源代码清单

|  |  |
| --- | --- |
| CourseDesign.pro | 工程配置文件，包括所需要加载的模块和相关的链接关系 |
| Personaltrip.h | 结构体声明，链表相关操作核心代码声明 |
| Personaltrip.cpp | 链表相关操作的实现 |
| newRoute.h | 新建行程相关窗体的声明 |
| newRoute.cpp | 新建行程相关窗体的实现 |
| newDayInfo.h | 新建每日信息相关窗体的声明 |
| newDayInfo.cpp | 新建每日信息相关窗体的声明 |
| newDayEXPInfo.h | 新建每日消费信息相关窗体的声明 |
| newDayEXPInfo.cpp | 新建每日消费信息相关窗体的声明 |
| mainwindow.h | 主窗体相关信号槽和相关控件，相关函数的声明 |
| mainwindow.cpp | 相关信号槽和函数的绑定，相关函数的实现 |
| main.cpp | 主函数体主窗体循环 |

### CourseDesign.pro

QT += core gui

QT += charts

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

TARGET = CourseDesign

TEMPLATE = app

DEFINES += QT\_DEPRECATED\_WARNINGS

SOURCES += \

main.cpp \

mainwindow.cpp \

newRoute.cpp \

PersonalTrip.cpp \

newDayInfo.cpp \

newDayEXPInfo.cpp

HEADERS += \

mainwindow.h \

newRoute.h \

Personaltrip.h \

newDayInfo.h \

newDayEXPInfo.h

FORMS += \

mainwindow.ui \

newRoute.ui \

newDayInfo.ui \

newDayEXPInfo.ui

### Personaltrip.h

#ifndef PERSONALTRIP\_H

#define PERSONALTRIP\_H

#pragma once

//进行相关结构体的声明

#include<string.h>

#include<stdio.h>

#include<stdlib.h>

#include<Windows.h>//其中内含wingdi

#include<wingdi.h>

#include<stdbool.h> //使得bool合法

#define \_CRT\_SECURE\_NO\_WARNINGS

typedef struct DressingIndex //穿衣指数结构体

{

int minT;

int maxT;

char recommendedDressing[20];//推荐的着装//出行期温度及穿衣指数

}DressingIndex;

typedef struct Route //行程一级链表结构体

{

int routeNum;//行程编号

char routeName[20];//主要目的地名称

char routeCountry[20];//目的地所属国家/省份

DressingIndex dressingIndex;

char bigTrans[20];//大交通 主要交通

char mainScene[200];//主要景点

char remoteNum[10];//异地办理手续行程编号 此处更改为char

//如需异地面签或提交材料，给出异地办手续的行程编号，否则为空

char beginDate[20];//出发日期

char arriveDate[20];//返回到达日期

int totalTime;//行程总天数

int totalNum;//行程总人数

float totalPrice;//行程总费用

struct DayInfo\* hDayInfo;//行程具体信息头指针

struct Route\* nextRoute;//链表next指针

}Route, \*p\_Route;

typedef struct DayInfo //行程具体信息结构体

{

int routeNum;//所属行程编号

int order;//序号 2代表第二天 存成int

char breakfastLoc[50];//早餐地点

char morningScene[50];//上午景点

char LunchLoc[50];//午餐地点

char afternoonScene[50];//下午景点

char dinnerLoc[50];//晚餐地点

char transInfo[200];//小交通

char residence[50];//住宿地点

struct DayEXPInfo\* hDayEXPInfo;

struct DayInfo\* nextDayInfo;//链表next指针

}DayInfo, \*p\_DayInfo;

typedef struct DayEXPInfo //每天消费信息

{

int routeNum;

int order;//序号 2代表第二天

int serialNum;//顺序号

char TxTime[20];//交易时间

char TxKind[16];//交易种类

float TxAmount;//交易金额

char isPrepayment[10];//是否预支付

char payment[10];//支付方式

char explain[50];//说明

struct DayEXPInfo\* nextDayEXPInfo;

}DayEXPInfo, \*p\_DayEXPInfo;

extern p\_Route hRoute;//声明为静态变量

//函数声明

bool creatRoute();//初始化链表

bool insertRoute(p\_Route\* curRoute); //当前节点

bool findRouteByNum(p\_Route cur, p\_Route\* pre, p\_Route\* tail);//用于插入 cur为需要查找是否重复的节点 pre用来记录前驱 tail用来记录整个链表尾部

p\_Route findRouteByNum(int num);//用于查询

bool printRoute();//打印以及链表

//打印整个链表

bool deleteRoute(int routeNum);

bool insertDayInfo(p\_Route\* route, p\_DayInfo\* cur);

bool insertDayInfo(p\_DayInfo\* cur);

bool findDayInfoByOrder(p\_Route route, p\_DayInfo cur, p\_DayInfo\* pre, p\_DayInfo\* tail);//删除第几天

p\_DayInfo findDayInfoByOrder(p\_Route\* route, int order);// 二级链表 通过和行程编号序号找到 用于查询

p\_DayInfo findDayInfoByOrder(int routeNum, int order);// 二级链表 通过和行程编号序号找到 用于查询 用节点的封装

bool printDayInfo(p\_Route route);//打印二级链表

bool deleteDayInfo(int routeNum, int order);//单个删除

bool insertDayEXPInfo(p\_Route\* route, p\_DayInfo\* dayinfo,p\_DayEXPInfo\* cur);

bool insertDayEXPInfo(p\_DayEXPInfo\* cur);

bool findDayEXPInfoByserialNum(p\_DayInfo dayinfo, p\_DayEXPInfo cur, p\_DayEXPInfo\* pre, p\_DayEXPInfo\* tail);//用于插入

p\_DayEXPInfo findDayEXPInfoByserialNum(int routeNum, int order, int serialNum);//用于查询

bool printDayEXPInfo(int routeNum, int order, int serialNum);//打印三级链表

bool deleteDayEXPInfo(int routeNum, int order, int serialNum);//单个删除

//save opreation

bool saveToFile();

bool loadFromFile();

bool printAll();

void sortRouteByNum(p\_Route \* pHead);//由于没有头结点，所以链表的头结点可能被更改

void sawpRoutePf(p\_Route A, p\_Route B,p\_Route\* head);

int GetRouteLength(p\_Route pHead);

void sortDayInfoByOrder(p\_DayInfo \* pHead);//由于没有头结点，所以链表的头结点可能被更改

void sawpDayInfoPf(p\_DayInfo A, p\_DayInfo B,p\_DayInfo\* head);

int GetDayInfoLength(p\_DayInfo pHead);

void sortDayEXPInfoBySerial(p\_DayEXPInfo \* pHead);//由于没有头结点，所以链表的头结点可能被更改

void sawpDayEXPInfoPf(p\_DayEXPInfo A, p\_DayEXPInfo B,p\_DayEXPInfo\* head);

int GetDayEXPInfoLength(p\_DayEXPInfo pHead);

p\_DayInfo findTransInfoByScene(char Scene[50]);

float findEXPByResidence(char Residence[50]);

#endif // PERSONALTRIP\_H

### PersonalTrip.cpp

#include"PersonalTrip.h"

#include"mainwindow.h"

p\_Route hRoute=NULL;

bool creatRoute()//初始化链表

{

hRoute=(p\_Route)malloc(sizeof(Route));

hRoute->nextRoute=NULL;

return true;

}

bool insertRoute(p\_Route\* curRoute) //当前节点

{

if(hRoute==NULL)

{

creatRoute();

hRoute = \*curRoute;

(\*curRoute)->nextRoute = NULL;

(\*curRoute)->hDayInfo = NULL;

return true;

}

p\_Route pre=NULL;

p\_Route tail = NULL;

if(findRouteByNum(\*curRoute,&pre,&tail))//通过行程编号删除

{

QMessageBox msgBox;

msgBox.setText("当前节点已经存在");

msgBox.setInformativeText("请直接修改");

msgBox.*exec*();

return false;

}

else

{

(tail)->nextRoute=\*curRoute;

(\*curRoute)->nextRoute= NULL;

(\*curRoute)->hDayInfo = NULL;

}

return true;

}

//查找当前链表的前置位 然后插入

bool deleteRoute(int routeNum)//reuturn 的问题

{

p\_Route p = hRoute;

if (p == NULL)

{

return false;

}

if (p->routeNum == routeNum)//头节点情况下

{

hRoute = p->nextRoute; //一级节点指针处理

//处理二级节点

p\_DayInfo p2 = p->hDayInfo;

p\_DayInfo temp2;

while (p2 != NULL)

{

temp2 = p2;

//处理三级节点

p\_DayEXPInfo p3=p2->hDayEXPInfo;

p\_DayEXPInfo temp3;

while (p3 != NULL)

{

temp3 = p3;

p3 = p3->nextDayEXPInfo;

free(temp3);

}

p2 = p2->nextDayInfo;

free(temp2);

}

free(p);

return true;

}

else

{

p\_Route pre;

while (p->routeNum!=routeNum)

{

pre = p;//记录p的前驱

p = p->nextRoute;

if (p == NULL)

{

return false;//没有找到

}

}

pre->nextRoute = p->nextRoute;//一级节点指针处理

//二级节点处理

p\_DayInfo p2 = p->hDayInfo;

p\_DayInfo temp2;

while (p2 != NULL)

{

temp2 = p2;

//处理三级节点

p\_DayEXPInfo p3 = p2->hDayEXPInfo;

p\_DayEXPInfo temp3;

while (p3 != NULL)

{

temp3 = p3;

p3 = p3->nextDayEXPInfo;

free(temp3);

}

p2 = p2->nextDayInfo;

free(temp2);

}

free(p);

return true;

}

}

//通过行程编号找到

bool findRouteByNum(p\_Route cur,p\_Route\* pre, p\_Route\* tail)//cur为需要查找是否重复的节点 pre用来记录前驱 tail用来记录整个链表尾部 返回的bool

{

p\_Route p=hRoute;//指向头结点 默认不为空的额情况下来找

\*tail=p;//记录p的前驱

while (p != NULL)//需要改变指针的指向 需要传二级指针

{

if (p->routeNum==cur->routeNum)

{

\*pre=\*tail;

return true;

}

\*tail=p;//记录p的前驱

p = p->nextRoute;//NULL这里出的问题

}

return false;

}

p\_Route findRouteByNum(int num)//返回所要找的route 用于查询

{

p\_Route p = hRoute;//指向头结点 默认不为空的额情况下来找

while (p != NULL)//需要改变指针的指向 需要传二级指针

{

if (p->routeNum == num)

{

return p;

}

p = p->nextRoute;

}

return NULL;

}

bool printRoute()

{

p\_Route p=hRoute;

if(hRoute==NULL)

{

return false;

}

else

{

p = p->nextRoute;

while (p != NULL)

{

printf("%d\n", p->routeNum);

p = p->nextRoute;

}

return true;

}

}

bool findDayInfoByOrder(p\_Route route, p\_DayInfo cur, p\_DayInfo\* pre, p\_DayInfo\* tail)//删除第几天

{

if (route == NULL)

{

printf("一级节点不存在");

return false;

}

if (route->hDayInfo == NULL)

{

printf("二级链表为空");

}

p\_DayInfo p = route->hDayInfo;

while (p != NULL)//需要改变指针的指向 需要传二级指针

{

if (p->order==cur->order)

{

\*pre = \*tail;

return true;

}

\*tail = p;//记录p的前驱

p = p->nextDayInfo;

}

return false;

}

bool insertDayInfo(p\_Route\* route, p\_DayInfo\* cur)

{

p\_DayInfo p = (\*route)->hDayInfo;

if (p == NULL)

{

(\*route)->hDayInfo = \*cur;

(\*cur)->nextDayInfo = NULL;

(\*cur)->hDayEXPInfo = NULL;

return true;

}

p\_DayInfo pre=(\*route)->hDayInfo;//改变这两个

p\_DayInfo tail=(\*route)->hDayInfo;

if (findDayInfoByOrder(\*route, \*cur, &pre, &tail))

{

//是否覆盖

QMessageBox msgBox;

msgBox.setText("当前节点已经存在");

msgBox.setInformativeText("请直接修改");

msgBox.*exec*();

return false;

}

else

{

(tail)->nextDayInfo = \*cur;

(\*cur)->nextDayInfo = NULL;

(\*cur)->hDayEXPInfo = NULL;

return true;

}

}

bool insertDayInfo(p\_DayInfo\* cur)

{

p\_Route p1 = findRouteByNum((\*cur)->routeNum);

if(p1==NULL)

{

return false;

}

p\_DayInfo p2=p1->hDayInfo;

if (p2 == NULL)

{

p1->hDayInfo = \*cur;//改变的是p1；

p2=p1->hDayInfo;

(\*cur)->nextDayInfo = NULL;

(\*cur)->hDayEXPInfo = NULL;

return true;

}

p\_DayInfo pre=p1->hDayInfo;//改变这两个

p\_DayInfo tail=p1->hDayInfo;

if (findDayInfoByOrder(p1, \*cur, &pre, &tail))

{

//如果找到是否覆盖

QMessageBox msgBox;

msgBox.setText("当前节点已经存在");

msgBox.setInformativeText("请直接修改");

msgBox.*exec*();

return false;

}

else

{

(tail)->nextDayInfo = \*cur;

(\*cur)->nextDayInfo = NULL;

(\*cur)->hDayEXPInfo = NULL;

return true;

}

}

bool printDayInfo(p\_Route route)

{

if (route == NULL)

{

printf("Error\n");

return false;

}

if (route->hDayInfo == NULL)

{

printf("Error\n");

return false;

}

p\_DayInfo p = route->hDayInfo;

while (p!=NULL)

{

printf("%d",p->order);

p = p->nextDayInfo;

if (p != NULL)

{

printf("->");

}

}

printf("\n");

return true;

}

p\_DayInfo findDayInfoByOrder(p\_Route\* route, int order)// 二级链表 通过和行程编号序号找到 用于查询

{

if (\*route == NULL)

{

return NULL;

}

if ((\*route)->hDayInfo == NULL)

{

return NULL;

}

p\_DayInfo p = (\*route)->hDayInfo;

while (p!=NULL)

{

if (p->order == order)

{

return p;

}

p = p->nextDayInfo;

}

return NULL;

}

p\_DayInfo findDayInfoByOrder(int routeNum, int order)// 二级链表 通过和行程编号序号找到 用于查询

{

p\_Route route = findRouteByNum(routeNum);

if (route == NULL)

{

return NULL;

}

p\_DayInfo dayinfo = findDayInfoByOrder(&route, order);

if (dayinfo == NULL)

{

return NULL;

}

return dayinfo;

}

bool deleteDayInfo(int routeNum, int order)//单个删除

{

//我要找前驱

p\_Route route = findRouteByNum(routeNum);

if (route == NULL)

{

return NULL;

}

p\_DayInfo p = route->hDayInfo;

if (p->order == order)//删除头结点

{

if (p->nextDayInfo == NULL)

{

route->hDayInfo = NULL;

//处理三级节点

p\_DayEXPInfo temp;

p\_DayEXPInfo p3;

p3 = p->hDayEXPInfo;

while (p3 != NULL)

{

temp = p3; //那么它存在的必要是什么呢？

p3 = p3->nextDayEXPInfo;

free(temp);

}

free(p);

return true;

}

else

{

route->hDayInfo = p->nextDayInfo;

//处理三级节点

p\_DayEXPInfo temp;

p\_DayEXPInfo p3;

p3 = p->hDayEXPInfo;

while (p3 != NULL)

{

temp = p3; //那么它存在的必要是什么呢？

p3 = p3->nextDayEXPInfo;

free(temp);

}

free(p);

return true;

}

}

else

{

p\_DayInfo pre;

while (p->order != order)

{

if (p->nextDayInfo == NULL)

{

return false;//没找到

}

pre = p;//记录前驱

p = p->nextDayInfo;

}

pre->nextDayInfo = p->nextDayInfo;

//处理三级节点

p\_DayEXPInfo temp;

p\_DayEXPInfo p3;

p3 = p->hDayEXPInfo;

while (p3 != NULL)

{

temp = p3; //那么它存在的必要是什么呢？

p3 = p3->nextDayEXPInfo;

free(temp);

}

free(p);

return true;

}

}

bool insertDayEXPInfo(p\_Route\* route, p\_DayInfo\* dayinfo, p\_DayEXPInfo\* cur)//需要写一些重载

{

if (\*dayinfo == NULL)

{

return false;

}

if ((\*dayinfo)->hDayEXPInfo == NULL)

{

(\*dayinfo)->hDayEXPInfo = \*cur;

(\*cur)->nextDayEXPInfo = NULL;

return true;

}

p\_DayEXPInfo pre;//改变这两个

p\_DayEXPInfo tail;

if (findDayEXPInfoByserialNum(\*dayinfo, \*cur, &pre, &tail))

{

//是否覆盖

QMessageBox msgBox;

msgBox.setText("当前节点已经存在");

msgBox.setInformativeText("请直接修改");

msgBox.*exec*();

return false;

}

else

{

tail->nextDayEXPInfo = \*cur;

(\*cur)->nextDayEXPInfo = NULL;

return true;

}

}

bool findDayEXPInfoByserialNum(p\_DayInfo dayinfo, p\_DayEXPInfo cur, p\_DayEXPInfo\* pre, p\_DayEXPInfo\* tail)//用于插入和删除

{

if (dayinfo == NULL)

{

printf("");

return false;

}

if (dayinfo->hDayEXPInfo == NULL)//判定不为空才找的

{

printf("三级链表为空");

}

p\_DayEXPInfo p = dayinfo->hDayEXPInfo;

while (p != NULL)//需要改变指针的指向 需要传二级指针

{

if (p->serialNum == cur->serialNum)

{

\*pre = \*tail;

return true;

}

\*tail = p;//记录p的前驱

p = p->nextDayEXPInfo;

}

return false;

}

p\_DayEXPInfo findDayEXPInfoByserialNum(int routeNum,int order,int serialNum)//用于查询

{

p\_Route route = findRouteByNum(routeNum);

if (route == NULL)

{

return NULL;

}

p\_DayInfo dayinfo = findDayInfoByOrder(&route, order);

if (dayinfo == NULL)

{

return NULL;

}

p\_DayEXPInfo p = dayinfo->hDayEXPInfo;

while (p != NULL)//需要改变指针的指向 需要传二级指针

{

if (p->serialNum == serialNum)

{

return p;

}

p = p->nextDayEXPInfo;

}

return NULL;

}

bool printDayEXPInfo(int routeNum, int order, int serialNum)//打印三级链表

{

p\_DayEXPInfo p = findDayEXPInfoByserialNum(routeNum, order, serialNum);

if (p == NULL)

{

return false;

}

//输出信息

printf("%d\n", p->routeNum);

printf("%d\n", p->order);

printf("%d\n", p->serialNum);

return true;

}

bool deleteDayEXPInfo(int routeNum, int order, int serialNum)//单个删除

{

//我要找前驱

p\_Route route = findRouteByNum(routeNum);

if (route == NULL)

{

return NULL;

}

p\_DayInfo dayinfo = findDayInfoByOrder(&route, order);

if (dayinfo == NULL)

{

return NULL;

}

p\_DayEXPInfo p = dayinfo->hDayEXPInfo;

if (p->serialNum == serialNum)//删除头结点

{

if (p->nextDayEXPInfo == NULL)

{

dayinfo->hDayEXPInfo = NULL;

free(p);

}

else

{

dayinfo->hDayEXPInfo = p->nextDayEXPInfo;

free(p);

}

return true;

}

else

{

p\_DayEXPInfo pre;

while (p->serialNum != serialNum)

{

if (p->nextDayEXPInfo != NULL)

{

return false;

}

pre = p;//记录前驱

p = p->nextDayEXPInfo;

}

pre->nextDayEXPInfo = p->nextDayEXPInfo;

free(p);

return true;

}

}

bool printAll()

{

p\_Route p1= hRoute;

if (hRoute == NULL)

{

return false;

}

while (p1!=NULL)

{

printf("%d\n",p1->routeNum);

p\_DayInfo p2 = p1->hDayInfo;

while (p2!=NULL)

{

printf("\t%d\n", p2->order);

p\_DayEXPInfo p3 = p2->hDayEXPInfo;

while (p3 != NULL)

{

printf("\t\t%d\n", p3->serialNum);

p3 = p3->nextDayEXPInfo;

}

p2 = p2->nextDayInfo;

}

p1 = p1->nextRoute;

}

return true;

}

//再封装一次

bool insertDayEXPInfo(p\_DayEXPInfo\* cur)

{

p\_Route p1 = findRouteByNum((\*cur)->routeNum);

if (p1 == NULL)

{

return false;

}

p\_DayInfo p2 = findDayInfoByOrder(&p1,(\*cur)->order);

if (p2 == NULL)

{

return false;

}

if (insertDayEXPInfo(&p1,&p2, cur))

{

return true;

}

else

{

return false;

}

}

bool saveToFile()

{

FILE \*out1, \*out2, \*out3;

p\_Route p1 = hRoute;// load head node

p\_DayInfo p2 = NULL;

p\_DayEXPInfo p3 = NULL;

if ((out1 = fopen("Data\\Route.dat", "wb")) == NULL)

return false;

if ((out2 = fopen("Data\\DayInfo.dat", "wb")) == NULL)

return false;

if ((out3 = fopen("Data\\DayEXPInfo.dat", "wb")) == NULL)

return false;

while (p1 != NULL)

{

fwrite(p1, sizeof(Route), 1, out1);

p2 = p1->hDayInfo;

while (p2)

{

fwrite(p2, sizeof(DayInfo), 1, out2);

p3 = p2->hDayEXPInfo;

while (p3)

{

fwrite(p3, sizeof(DayEXPInfo), 1, out3);

p3 = p3->nextDayEXPInfo;

}

p2 = p2->nextDayInfo;

}

p1 = p1->nextRoute;

}

fclose(out1);

fclose(out2);

fclose(out3);

return true;

}

bool loadFromFile()

{

FILE\*in1, \*in2, \*in3;

p\_Route p1 = NULL;

p\_DayInfo p2;

p\_DayEXPInfo p3;

if ((in1 = fopen("Data\\Route.dat", "rb")) == NULL)

return false;

if ((in2 = fopen("Data\\DayInfo.dat", "rb")) == NULL)

return false;

if ((in3 = fopen("Data\\DayExpInfo.dat", "rb")) == NULL)

return false;

p\_Route pre\_p1=NULL;

bool isFirst=true;

while (!feof(in1))//load the first-class list

{

if(isFirst)

{

p1 = (p\_Route)malloc(sizeof(Route));

p1->routeNum=-1;

fread(p1, sizeof(Route), 1, in1);

if(p1->routeNum==-1)

{

free(p1);

break;

}

p1->hDayInfo=NULL;

p1->nextRoute = NULL;

hRoute=p1;// the head pointer is aimed at the p1

pre\_p1=p1;

isFirst=false;

}

else

{

p\_Route p1 = (p\_Route)malloc(sizeof(Route));

p1->routeNum=-1;

fread(p1, sizeof(Route), 1, in1);

if(p1->routeNum==-1)

{

free(p1);

break;

}

p1->hDayInfo=NULL;

p1->nextRoute = NULL;

pre\_p1->nextRoute=p1;

pre\_p1=p1;

}

}

while (!feof(in2))/\*将存储站点链的文件调入内存\*/

{

p2 = (p\_DayInfo)malloc(sizeof(DayInfo));

p2->order=-1;

fread(p2, sizeof(DayInfo), 1, in2);

if(p2->order==-1)

{

free(p2);

break;

}

p2->nextDayInfo=NULL;

p2->hDayEXPInfo=NULL;

p1=hRoute;

while (p1 != NULL)

{

if (p1->routeNum==p2->routeNum)

{

if(p1->hDayInfo==NULL)

{

p1->hDayInfo=p2;

break;

}

else

{

p\_DayInfo tp2=p1->hDayInfo;

p\_DayInfo pre=p1->hDayInfo;

while(tp2->nextDayInfo!=NULL)

{

pre=tp2;

tp2=tp2->nextDayInfo;

}

pre->nextDayInfo=p2;

break;

}

}

else

{

p1=p1->nextRoute;

}

}

}

while (!feof(in3))/\*将存储车辆信息链的文件调入内存\*/

{

p3 = (p\_DayEXPInfo)malloc(sizeof(DayEXPInfo));

p3->serialNum=-1;

fread(p3, sizeof(DayEXPInfo), 1, in3);

if(p3->serialNum==-1)

{

free(p3);

break;

}

p3->nextDayEXPInfo=NULL;

p1=hRoute;

while (p1 != NULL)

{

if (p1->routeNum==p3->routeNum)

{

p\_DayInfo tp2=p1->hDayInfo;

if(tp2->order==p3->order)

{

p\_DayEXPInfo tp3=tp2->hDayEXPInfo;

if(tp3==NULL)

{

tp2->hDayEXPInfo=p3;

break;

}

else

{

if(tp3->nextDayEXPInfo==NULL)

{

tp3->nextDayEXPInfo=p3;

}

else

{

p\_DayEXPInfo pre=NULL;

while(tp3->nextDayEXPInfo!=NULL)

{

pre=tp3;

tp3=tp3->nextDayEXPInfo;

}

pre->nextDayEXPInfo=p3;

break;

}

}

}

else

{

tp2=tp2->nextDayInfo;

if(tp2==NULL)

{

return false;

}

}

}

p1=p1->nextRoute;

}

}

return true;

}

p\_DayInfo findTransInfoByScene(char Scene[50])

{

p\_Route p1=hRoute;

if(hRoute==NULL)

{

return NULL;

}

while(p1!=NULL)

{

p\_DayInfo p2=p1->hDayInfo;

while(p2!=NULL)

{

if(strcmp(p2->morningScene, Scene)==0||strcmp(p2->afternoonScene, Scene)==0)

{

return p2;

}

p2=p2->nextDayInfo;

}

p1=p1->nextRoute;

}

return NULL;

}

float findEXPByResidence(char Residence[50])

{

p\_Route p1=hRoute;

float residenceEXP=-1;

if(hRoute==NULL)

{

return residenceEXP;

}

while(p1!=NULL)

{

p\_DayInfo p2=p1->hDayInfo;

while(p2!=NULL)

{

if(strcmp(p2->residence, Residence)==0)

{

p\_DayEXPInfo p3=p2->hDayEXPInfo;

while(p3!=NULL)

{

if(strcmp(p3->TxKind, "住宿")==0)

{

residenceEXP+=p3->TxAmount;

}

p3=p3->nextDayEXPInfo;

}

}

p2=p2->nextDayInfo;

}

p1=p1->nextRoute;

}

return residenceEXP;

}

bool sortByRouteNum()//quick sort

{

p\_Route p1=hRoute;

p\_Route pre\_p1=hRoute;

if(p1==NULL)

{

return false;

}

while(p1->nextRoute != NULL)

{

p\_Route p;

p\_Route pt=p1->nextRoute;

if((p1->routeNum)>(pt->routeNum))

{

p\_Route p2=p1->nextRoute;

p\_Route p3=p1->nextRoute;

pre\_p1->nextRoute=p2;

p2->nextRoute=p1;

p1->nextRoute=p3;

pre\_p1=p2;

}

p1=p1->nextRoute; //p再前进一个节点

pre\_p1=pre\_p1->nextRoute;

}

return true;

}

void sortRouteByNum(p\_Route \* pHead)//由于没有头结点，所以链表的头结点可能被更改

{

if((\* pHead)==NULL)

{

return;

}

int length = GetRouteLength(\*pHead);

p\_Route pMove=(\*pHead);

for (int i = 0; i < length; i++) //需要进行(n-1)次遍历,控制次数

{

if(pMove!=NULL)

{

while (pMove!=NULL&&pMove->nextRoute != NULL)

{

if (pMove->routeNum > pMove->nextRoute->routeNum)

{

sawpRoutePf(pMove, pMove->nextRoute,pHead);

printRoute();

}

pMove = pMove->nextRoute;

}

}

//每次遍历结束，pMove重新移动到链表头部

pMove=(\*pHead);

}

}

int GetRouteLength(p\_Route pHead)

{

int Length = 0;

while (pHead != NULL)

{

++Length;

pHead = pHead->nextRoute;

}

return Length;

}

void sawpRoutePf(p\_Route A, p\_Route B,p\_Route\* head)

{

p\_Route t\_head=(\*head);

p\_Route temp;

p\_Route pre\_A;

p\_Route pre\_B;

if(A->routeNum==(\*head)->routeNum)

{

if(A->nextRoute==B)

{

(\*head)=B;//B的前指针

A->nextRoute=B->nextRoute;

B->nextRoute=A;

return;

}

while (t\_head != NULL)

{

if (t\_head->nextRoute == A)

{

pre\_A = t\_head;

}

if (t\_head->nextRoute == B)

{

pre\_B = t\_head;

}

t\_head = t\_head->nextRoute;

}

temp = B->nextRoute;

(\*head)=B;//B的前指针

B->nextRoute=A->nextRoute;//B的后指针

pre\_B->nextRoute=A;//A的前指针

A->nextRoute = temp;

}

while (t\_head != NULL)

{

if (t\_head->nextRoute == A)

{

pre\_A = t\_head;

}

if (t\_head->nextRoute == B)

{

pre\_B = t\_head;

}

t\_head = t\_head->nextRoute;

}

if(A->nextRoute==B)

{

pre\_A->nextRoute = B;//B的前节点

A->nextRoute=B->nextRoute;

B->nextRoute=A;

return;

}

temp = B->nextRoute;

pre\_A->nextRoute = B;//B的前节点

B->nextRoute = A->nextRoute;//B的后节点

pre\_B->nextRoute = A;//A的前节点

A->nextRoute = temp;//A的前节点

}

void sortDayInfoByOrder(p\_DayInfo \* pHead)//由于没有头结点，所以链表的头结点可能被更改

{

if((\* pHead)==NULL)

{

return;

}

int length = GetDayInfoLength(\*pHead);

p\_DayInfo pMove=(\*pHead);

for (int i = 0; i < length; i++) //需要进行(n-1)次遍历,控制次数

{

if(pMove!=NULL)

{

while (pMove!=NULL&&pMove->nextDayInfo != NULL)

{

if (pMove->order > pMove->nextDayInfo->order)

{

sawpDayInfoPf(pMove, pMove->nextDayInfo,pHead);

printRoute();

}

pMove = pMove->nextDayInfo;

}

}

//每次遍历结束，pMove重新移动到链表头部

pMove=(\*pHead);

}

}

int GetDayInfoLength(p\_DayInfo pHead)

{

int Length = 0;

while (pHead != NULL)

{

++Length;

pHead = pHead->nextDayInfo;

}

return Length;

}

void sawpDayInfoPf(p\_DayInfo A, p\_DayInfo B,p\_DayInfo\* head)

{

p\_DayInfo t\_head=(\*head);

p\_DayInfo temp;

p\_DayInfo pre\_A;

p\_DayInfo pre\_B;

if(A->order==(\*head)->order)

{

if(A->nextDayInfo==B)

{

(\*head)=B;//B的前指针

A->nextDayInfo=B->nextDayInfo;

B->nextDayInfo=A;

return;

}

while (t\_head != NULL)

{

if (t\_head->nextDayInfo == A)

{

pre\_A = t\_head;

}

if (t\_head->nextDayInfo == B)

{

pre\_B = t\_head;

}

t\_head = t\_head->nextDayInfo;

}

temp = B->nextDayInfo;

(\*head)=B;//B的前指针

B->nextDayInfo=A->nextDayInfo;//B的后指针

pre\_B->nextDayInfo=A;//A的前指针

A->nextDayInfo = temp;

}

while (t\_head != NULL)

{

if (t\_head->nextDayInfo == A)

{

pre\_A = t\_head;

}

if (t\_head->nextDayInfo == B)

{

pre\_B = t\_head;

}

t\_head = t\_head->nextDayInfo;

}

if(A->nextDayInfo==B)

{

pre\_A->nextDayInfo = B;//B的前节点

A->nextDayInfo=B->nextDayInfo;

B->nextDayInfo=A;

return;

}

temp = B->nextDayInfo;

pre\_A->nextDayInfo = B;//B的前节点

B->nextDayInfo = A->nextDayInfo;//B的后节点

pre\_B->nextDayInfo = A;//A的前节点

A->nextDayInfo = temp;//A的前节点

printf("\n");

}

void sortDayEXPInfoBySerial(p\_DayEXPInfo \* pHead)//由于没有头结点，所以链表的头结点可能被更改

{

if((\* pHead)==NULL)

{

return;

}

int length = GetDayEXPInfoLength(\*pHead);

p\_DayEXPInfo pMove=(\*pHead);

for (int i = 0; i < length; i++) //需要进行(n-1)次遍历,控制次数

{

if(pMove!=NULL)

{

while (pMove!=NULL&&pMove->nextDayEXPInfo != NULL)

{

if (pMove->serialNum > pMove->nextDayEXPInfo->serialNum)

{

sawpDayEXPInfoPf(pMove, pMove->nextDayEXPInfo,pHead);

printRoute();

}

pMove = pMove->nextDayEXPInfo;

}

}

//每次遍历结束，pMove重新移动到链表头部

pMove=(\*pHead);

}

}

int GetDayEXPInfoLength(p\_DayEXPInfo pHead)

{

int Length = 0;

while (pHead != NULL)

{

++Length;

pHead = pHead->nextDayEXPInfo;

}

return Length;

}

void sawpDayEXPInfoPf(p\_DayEXPInfo A, p\_DayEXPInfo B,p\_DayEXPInfo\* head)

{

p\_DayEXPInfo t\_head=(\*head);

p\_DayEXPInfo temp;

p\_DayEXPInfo pre\_A;

p\_DayEXPInfo pre\_B;

if(A->serialNum==(\*head)->serialNum)

{

if(A->nextDayEXPInfo==B)

{

(\*head)=B;//B的前指针

A->nextDayEXPInfo=B->nextDayEXPInfo;

B->nextDayEXPInfo=A;

return;

}

while (t\_head != NULL)

{

if (t\_head->nextDayEXPInfo == A)

{

pre\_A = t\_head;

}

if (t\_head->nextDayEXPInfo == B)

{

pre\_B = t\_head;

}

t\_head = t\_head->nextDayEXPInfo;

}

temp = B->nextDayEXPInfo;

(\*head)=B;//B的前指针

B->nextDayEXPInfo=A->nextDayEXPInfo;//B的后指针

pre\_B->nextDayEXPInfo=A;//A的前指针

A->nextDayEXPInfo = temp;

}

while (t\_head != NULL)

{

if (t\_head->nextDayEXPInfo == A)

{

pre\_A = t\_head;

}

if (t\_head->nextDayEXPInfo == B)

{

pre\_B = t\_head;

}

t\_head = t\_head->nextDayEXPInfo;

}

if(A->nextDayEXPInfo==B)

{

pre\_A->nextDayEXPInfo = B;//B的前节点

A->nextDayEXPInfo=B->nextDayEXPInfo;

B->nextDayEXPInfo=A;

return;

}

temp = B->nextDayEXPInfo;

pre\_A->nextDayEXPInfo = B;//B的前节点

B->nextDayEXPInfo = A->nextDayEXPInfo;//B的后节点

pre\_B->nextDayEXPInfo = A;//A的前节点

A->nextDayEXPInfo = temp;//A的前节点

printf("\n");

}

### newRoute.h

#ifndef NEWROUTE\_H

#define NEWROUTE\_H

#include <QDialog>

#include<QMessageBox>

#include<QCloseEvent>

#include<QDialog>

#include"PersonalTrip.h"

#include"mainwindow.h"

namespace Ui

{

class newRoute;

}

class newRoute : public QDialog

{

Q\_OBJECT

public:

explicit newRoute(QWidget \*parent = 0);

~*newRoute*();

private slots:

void on\_saveRouteButton\_clicked();

private:

Ui::newRoute \*ui;

protected:

//void closeEvent(QCloseEvent \*event);

};

#endif // NEWROUTE\_H

### newRoute.cpp

#include "newRoute.h"

#include "ui\_newRoute.h"

#include"mainwindow.h"

newRoute::newRoute(QWidget \*parent) :

QDialog(parent),

ui(new Ui::newRoute)

{

ui->setupUi(this);

}

newRoute::~*newRoute*()

{

delete ui;

}

void newRoute::on\_saveRouteButton\_clicked()

{

p\_Route cur = (p\_Route)malloc(sizeof(Route)); //字符串赋值

cur->routeNum=ui->lineEdit->text().toInt();//int

strcpy(cur->routeName, ui->lineEdit\_2->text().toStdString().c\_str());

strcpy(cur->routeCountry, ui->lineEdit\_3->text().toStdString().c\_str());

cur->dressingIndex.minT=ui->lineEdit\_4->text().toInt();

cur->dressingIndex.maxT=ui->lineEdit\_5->text().toInt();

strcpy(cur->dressingIndex.recommendedDressing, ui->lineEdit\_6->text().toStdString().c\_str());

strcpy(cur->bigTrans, ui->lineEdit\_7->text().toStdString().c\_str());

strcpy(cur->mainScene, ui->lineEdit\_8->text().toStdString().c\_str());

strcpy(cur->remoteNum, ui->lineEdit\_9->text().toStdString().c\_str());

strcpy(cur->beginDate, ui->lineEdit\_10->text().toStdString().c\_str());

strcpy(cur->arriveDate, ui->lineEdit\_11->text().toStdString().c\_str());

cur->totalTime=ui->lineEdit\_12->text().toInt();//int 这个也是输入的

cur->totalNum=ui->lineEdit\_13->text().toInt();//int

cur->totalPrice=ui->lineEdit\_14->text().toFloat();//float

QMessageBox msgBox;

if(insertRoute(&cur))

{

msgBox.setText("Save Successfully");

msgBox.*exec*();

}

else

{

msgBox.setText("Save Fail!!!");

msgBox.*exec*();

}

this->close();

}

### newDayInfo.h

#ifndef NEWDAYINFO\_H

#define NEWDAYINFO\_H

#include <QDialog>

#include<QMessageBox>

#include"Personaltrip.h"

namespace Ui {

class newDayInfo;

}

class newDayInfo : public QDialog

{

Q\_OBJECT

public:

explicit newDayInfo(QWidget \*parent = 0);

~*newDayInfo*();

private slots:

void on\_saveDayInfoButton\_clicked();

private:

Ui::newDayInfo \*ui;

};

#endif // NEWDAYINFO\_H

### newDayInfo.cpp

#include "newDayInfo.h"

#include "ui\_newDayInfo.h"

#include"Personaltrip.h"

newDayInfo::newDayInfo(QWidget \*parent) :

QDialog(parent),

ui(new Ui::newDayInfo)

{

ui->setupUi(this);

this->setWindowTitle("插入每日信息");

}

newDayInfo::~*newDayInfo*()

{

delete ui;

}

void newDayInfo::on\_saveDayInfoButton\_clicked()

{

p\_DayInfo cur= (p\_DayInfo)malloc(sizeof(DayInfo));

cur->routeNum=ui->lineEdit->text().toInt();//int

cur->order=ui->lineEdit\_2->text().toInt();//int

strcpy(cur->breakfastLoc, ui->lineEdit\_3->text().toStdString().c\_str());

strcpy(cur->morningScene, ui->lineEdit\_4->text().toStdString().c\_str());

strcpy(cur->LunchLoc, ui->lineEdit\_5->text().toStdString().c\_str());

strcpy(cur->afternoonScene, ui->lineEdit\_6->text().toStdString().c\_str());

strcpy(cur->dinnerLoc, ui->lineEdit\_7->text().toStdString().c\_str());

strcpy(cur->transInfo, ui->lineEdit\_8->text().toStdString().c\_str());

strcpy(cur->residence, ui->lineEdit\_9->text().toStdString().c\_str());

QMessageBox msgBox;

if(insertDayInfo(&cur))

{

msgBox.setText("Save Successfully");

msgBox.*exec*();

}

else

{

msgBox.setText("Save fail");

msgBox.*exec*();

}

this->close();

}

### newDayEXPInfo.h

#ifndef NEWDAYEXPINFO\_H

#define NEWDAYEXPINFO\_H

#include <QDialog>

#include<QMessageBox>

#include"mainwindow.h"

#include"Personaltrip.h"

namespace Ui {

class newDayEXPInfo;

}

class newDayEXPInfo : public QDialog

{

Q\_OBJECT

public:

explicit newDayEXPInfo(QWidget \*parent = 0);

~*newDayEXPInfo*();

private slots:

void on\_saveDayEXPInfoButton\_clicked();

private:

Ui::newDayEXPInfo \*ui;

};

#endif // NEWDAYEXPINFO\_H

### newDayEXPInfo.cpp

#include "newDayEXPInfo.h"

#include "ui\_newDayEXPInfo.h"

newDayEXPInfo::newDayEXPInfo(QWidget \*parent) :

QDialog(parent),

ui(new Ui::newDayEXPInfo)

{

ui->setupUi(this);

this->setWindowTitle("插入消费信息");

}

newDayEXPInfo::~*newDayEXPInfo*()

{

delete ui;

}

//在插入的时候加

void newDayEXPInfo::on\_saveDayEXPInfoButton\_clicked()

{

//字符串赋值

p\_DayEXPInfo cur= (p\_DayEXPInfo)malloc(sizeof(DayEXPInfo));

cur->routeNum=ui->lineEdit->text().toInt();//int

cur->order=ui->lineEdit\_2->text().toInt();//int

int order=cur->order;

cur->serialNum=ui->lineEdit\_3->text().toInt();//int

strcpy(cur->TxTime, ui->lineEdit\_4->text().toStdString().c\_str());

strcpy(cur->TxKind, ui->lineEdit\_5->text().toStdString().c\_str());

cur->TxAmount=ui->lineEdit\_6->text().toFloat();//float

strcpy(cur->isPrepayment, ui->lineEdit\_7->text().toStdString().c\_str());

strcpy(cur->payment, ui->lineEdit\_8->text().toStdString().c\_str());

strcpy(cur->explain, ui->lineEdit\_9->text().toStdString().c\_str());

QMessageBox msgBox;

p\_Route p1=findRouteByNum(cur->routeNum);

if(p1==NULL)

{

msgBox.setText("routeNum error");

msgBox.*exec*();

this->close();

}

int routeNum=cur->routeNum;

p\_DayInfo p2=findDayInfoByOrder(routeNum,order);

if(p2==NULL)

{

msgBox.setText("order error");

msgBox.*exec*();

this->close();

}

if(insertDayEXPInfo(&p1,&p2,&cur))

{

msgBox.setText("Success");

msgBox.*exec*();

this->close();

}

else

{

msgBox.setText("fail");

msgBox.*exec*();

this->close();

}

}

### mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include<QMessageBox>

#include<QGroupBox>

#include<QTreeWidget>

#include<QTreeWidgetItem>

#include<QLabel>

#include<QDebug>

#include<QDialog>

#include<QLineEdit>

#include<QGridLayout>

#include<QPushButton>

#include<QDebug>

#include<string>

#include<QDateTime>

#include"newRoute.h"

#include"newDayInfo.h"

#include"newDayEXPInfo.h"

#include"personaltrip.h"

#include <QtWidgets>

#include <QtCharts>//use this to draw pie pict

#include <QPieSeries>

namespace Ui

{

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

//p\_Route hRoute;//

~*MainWindow*();

void showTree();

private slots:

//展示所有信息

void showAllslot();

//插入节点

void newRouteSlot();

void newDayInfoSlot();

void newDayEXPInfoSlot();

//获取所需删除信息

//Act表示执行删除操作

void deleteRouteSlot();

void Act\_deleteRouteSlot();

void deleteDayInfoSlot();

void Act\_deleteDayInfoSlot();

void deleteDayEXPInfoSlot();

void Act\_deleteDayEXPInfoSlot();

//查询操作

//其中Act表示执行显示查询结果

void showRouteslot();

void Act\_showRouteslot();

void showDayInfoslot();

void Act\_showDayInfoSlot();

void showDayEXPInfoslot();

void Act\_showDayEXPInfoSlot();

//文件操作

void saveToFileSlot();

void loadFromFileSlot();

//退出窗口操作

void quitBtnSLot();

//统计报表

void countEXPSlot();

void Mouse\_position(QTreeWidgetItem\* item);//获取当前鼠标所在的链表级别及具体节点顺序

void Show\_info(QTreeWidgetItem\* item,int n);//显示当前鼠标所在节点的信息

void showTransInfoSlot();//查询小交通信息

void Act\_showTransInfoSlot();

void showResidenceSlot();//查询住宿信息

void Act\_showResidenceSlot();

void showDressingIndexSlot();//查询着装信息

void Act\_showDressingIndexSlot();

//排序操作

void sortRouteSlot();

void sortDayInfoSlot();

void sortDayEXPInfoSlot();

void on\_modifyButton\_clicked();//执行修改操作

private:

Ui::MainWindow \*ui;

int now\_index[3];//now\_index to record the mouse position

QDialog \*newDialog;

QDialog \*deletedialog;

QDialog \*showDialog;

QLineEdit \*routeNumLineEdit;//不同槽之间需要调用

QLineEdit \*orderLineEdit;//不同槽之间需要调用

QLineEdit \*serialNumLineEdit;//不同槽之间需要调用

};

#endif // MAINWINDOW\_H

### mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "newRoute.h"

#include"newDayInfo.h"

#include"newDayEXPInfo.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

ui->treeWidget->setColumnCount(1);

ui->treeWidget->setHeaderLabel(tr("Route"));

ui->tableWidget->setColumnCount(2);

ui->tableWidget->setRowCount(15);

ui->tableWidget->setHorizontalHeaderLabels(QStringList() << "item" <<"information");

ui->tableWidget->verticalHeader()->*setVisible*(false);//hide vertical header

ui->tableWidget->setColumnWidth(0,200);

ui->tableWidget->setColumnWidth(1,200);

QTreeWidgetItem \*All = new QTreeWidgetItem(ui->treeWidget,QStringList(QString("All")));

//file

this->setWindowTitle("个人自助游管理信息系统\_Powered\_By\_Hover");

QObject::connect(ui->newRouteAction, SIGNAL(triggered()), this, SLOT(newRouteSlot()));

QObject::connect(ui->newDayInfoAction, SIGNAL(triggered()), this, SLOT(newDayInfoSlot()));

QObject::connect(ui->newDayEXPInfoAction, SIGNAL(triggered()), this, SLOT(newDayEXPInfoSlot()));

//QObject::connect(ui->exitAction,SIGNAL(triggered()),this,SLOT(close()));//是否判断未保存 人性化设计

//edit menu

QObject::connect(ui->deleteRouteAction, SIGNAL(triggered()), this, SLOT(deleteRouteSlot()));

QObject::connect(ui->deleteDayInfoAction, SIGNAL(triggered()), this, SLOT(deleteDayInfoSlot()));

QObject::connect(ui->deleteDayEXPInfoAction, SIGNAL(triggered()), this, SLOT(deleteDayEXPInfoSlot()));

QObject::connect(ui->showRouteAction, SIGNAL(triggered()), this, SLOT(showRouteslot()));

QObject::connect(ui->showDayInfoAction, SIGNAL(triggered()), this, SLOT(showDayInfoslot()));

QObject::connect(ui->showDayEXPInfoAction, SIGNAL(triggered()), this, SLOT(showDayEXPInfoslot()));

QObject::connect(ui->showAllAction, SIGNAL(triggered()), this, SLOT(showAllslot()));

QObject::connect(ui->saveToFileAction, SIGNAL(triggered()), this, SLOT(saveToFileSlot()));

QObject::connect(ui->loadFromFileAction, SIGNAL(triggered()), this, SLOT(loadFromFileSlot()));

QObject::connect(ui->treeWidget,SIGNAL(itemClicked(QTreeWidgetItem\*,int)),this,SLOT(Show\_info(QTreeWidgetItem\* ,int)));//mouse click -> show information

QObject::connect(ui->modifyRouteAction,SIGNAL(triggered()),this,SLOT(modifyRouteSlot()));

QObject::connect(ui->modifyDayInfoAction,SIGNAL(triggered()),this,SLOT(modifyDayInfoSlot()));

QObject::connect(ui->modifyDayEXPInfoAction,SIGNAL(triggered()),this,SLOT(modifyDayEXPInfoSlot()));

QObject::connect(ui->countEXPAction,SIGNAL(triggered()),this,SLOT(countEXPSlot()));

QObject::connect(ui->showTransInfoAction,SIGNAL(triggered()),this,SLOT(showTransInfoSlot()));

QObject::connect(ui->showResidenceAction,SIGNAL(triggered()),this,SLOT(showResidenceSlot()));

QObject::connect(ui->showDressingIndexAction,SIGNAL(triggered()),this,SLOT(showDressingIndexSlot()));

QObject::connect(ui->sortRouteAction,SIGNAL(triggered()),this,SLOT(sortRouteSlot()));

QObject::connect(ui->sortDayInfoAction,SIGNAL(triggered()),this,SLOT(sortDayInfoSlot()));

QObject::connect(ui->sortDayEXPInfoAction,SIGNAL(triggered()),this,SLOT(sortDayEXPInfoSlot()));

}

void MainWindow::Mouse\_position(QTreeWidgetItem\* item)//get mouse position use three int to store

{

now\_index[0] = -1;//initialization

now\_index[1] = -1;

now\_index[2] = -1;

int i = 0;

QTreeWidgetItem\* parent = item->parent();//get parent node

while(parent != NULL){//find all parents

now\_index[i] = parent->indexOfChild(item);

i++;

item = parent;

parent = item->parent();

}

// qDebug()<<now\_index[0]<<now\_index[1]<<now\_index[2];

}

void MainWindow::Show\_info(QTreeWidgetItem\* item,int n){//show details

Mouse\_position(item);//get mouse position

//nothing

if(now\_index[0] == -1 && now\_index[1] == -1 && now\_index[2] == -1)//未点击

{

return;

}

//route

if(now\_index[0] != -1 && now\_index[1] == -1 && now\_index[2] == -1)

{

p\_Route p1=hRoute;

//struct college\_info\* college\_temp = head;

for(int i = 0; i < now\_index[0]; ++i)

{//find position

p1 = p1->nextRoute;

}

if(p1 == NULL)

{

return ;

}

ui->tableWidget->setItem(0, 0, new QTableWidgetItem("行程编号"));//show information

ui->tableWidget->setItem(1, 0, new QTableWidgetItem("主要目的地"));

ui->tableWidget->setItem(2, 0, new QTableWidgetItem("目的地所属国家/省份"));

ui->tableWidget->setItem(3, 0, new QTableWidgetItem("最低温度"));

ui->tableWidget->setItem(4, 0, new QTableWidgetItem("最高温度"));

ui->tableWidget->setItem(5, 0, new QTableWidgetItem("推荐的着装"));

ui->tableWidget->setItem(6, 0, new QTableWidgetItem("大交通"));

ui->tableWidget->setItem(7, 0, new QTableWidgetItem("主要景点"));

ui->tableWidget->setItem(8, 0, new QTableWidgetItem("异地办理手续行程编号"));

ui->tableWidget->setItem(9, 0, new QTableWidgetItem("出发日期"));

ui->tableWidget->setItem(10, 0, new QTableWidgetItem("返回到达日期"));

ui->tableWidget->setItem(11, 0, new QTableWidgetItem("行程总天数"));

ui->tableWidget->setItem(12, 0, new QTableWidgetItem("行程总人数"));

ui->tableWidget->setItem(13, 0, new QTableWidgetItem("行程总费用"));

ui->tableWidget->setItem(0, 1, new QTableWidgetItem(QString::number(p1->routeNum, 10)));

ui->tableWidget->setItem(1, 1, new QTableWidgetItem(QString::fromUtf8(p1->routeName)));

ui->tableWidget->setItem(2, 1, new QTableWidgetItem(QString::fromUtf8(p1->routeCountry)));

ui->tableWidget->setItem(3, 1, new QTableWidgetItem(QString("%1").arg(p1->dressingIndex.minT)));

ui->tableWidget->setItem(4, 1, new QTableWidgetItem(QString("%1").arg(p1->dressingIndex.maxT)));

ui->tableWidget->setItem(5, 1, new QTableWidgetItem(QString::fromUtf8(p1->dressingIndex.recommendedDressing)));

ui->tableWidget->setItem(6, 1, new QTableWidgetItem(QString::fromUtf8(p1->bigTrans)));

ui->tableWidget->setItem(7, 1, new QTableWidgetItem(QString::fromUtf8(p1->mainScene)));

ui->tableWidget->setItem(8, 1, new QTableWidgetItem(QString::fromUtf8(p1->remoteNum)));

ui->tableWidget->setItem(9, 1, new QTableWidgetItem(QString::fromUtf8(p1->beginDate)));

ui->tableWidget->setItem(10, 1, new QTableWidgetItem(QString::fromUtf8(p1->arriveDate)));

ui->tableWidget->setItem(11, 1, new QTableWidgetItem(QString("%1").arg(p1->totalTime)));

ui->tableWidget->setItem(12, 1, new QTableWidgetItem(QString("%1").arg(p1->totalNum)));

ui->tableWidget->setItem(13, 1, new QTableWidgetItem(QString("%1").arg(p1->totalPrice)));

}

//DayInfo

if(now\_index[0] != -1 && now\_index[1] != -1 && now\_index[2] == -1)

{

p\_Route p1=hRoute;

for(int i = 0; i < now\_index[1]; ++i)//find position

{

p1 = p1->nextRoute;

}

p\_DayInfo p2 = p1->hDayInfo;

for(int i = 0; i < now\_index[0]; ++i)

{

p2 = p2->nextDayInfo;

}

if(p2 == NULL)

{

return ;

}

ui->tableWidget->setItem(0, 0, new QTableWidgetItem("所属行程编号"));//show information

ui->tableWidget->setItem(1, 0, new QTableWidgetItem("序号"));

ui->tableWidget->setItem(2, 0, new QTableWidgetItem("早餐地点"));

ui->tableWidget->setItem(3, 0, new QTableWidgetItem("上午景点"));

ui->tableWidget->setItem(4, 0, new QTableWidgetItem("午餐地点"));

ui->tableWidget->setItem(5, 0, new QTableWidgetItem("下午景点"));

ui->tableWidget->setItem(6, 0, new QTableWidgetItem("晚餐地点"));

ui->tableWidget->setItem(7, 0, new QTableWidgetItem("小交通"));

ui->tableWidget->setItem(8, 0, new QTableWidgetItem("住宿地点"));

ui->tableWidget->setItem(9, 0, new QTableWidgetItem(""));

ui->tableWidget->setItem(10, 0, new QTableWidgetItem(""));

ui->tableWidget->setItem(11, 0, new QTableWidgetItem(""));

ui->tableWidget->setItem(12, 0, new QTableWidgetItem(""));

ui->tableWidget->setItem(13, 0, new QTableWidgetItem(""));

ui->tableWidget->setItem(0, 1, new QTableWidgetItem(QString("%1").arg(p2->routeNum)));

ui->tableWidget->setItem(1, 1, new QTableWidgetItem(QString("%1").arg(p2->order)));

ui->tableWidget->setItem(2, 1, new QTableWidgetItem(QString::fromUtf8(p2->breakfastLoc)));

ui->tableWidget->setItem(3, 1, new QTableWidgetItem(QString::fromUtf8(p2->morningScene)));

ui->tableWidget->setItem(4, 1, new QTableWidgetItem(QString::fromUtf8(p2->LunchLoc)));

ui->tableWidget->setItem(5, 1, new QTableWidgetItem(QString::fromUtf8(p2->afternoonScene)));

ui->tableWidget->setItem(6, 1, new QTableWidgetItem(QString::fromUtf8(p2->dinnerLoc)));

ui->tableWidget->setItem(7, 1, new QTableWidgetItem(QString::fromUtf8(p2->transInfo)));

ui->tableWidget->setItem(8, 1, new QTableWidgetItem(QString::fromUtf8(p2->residence)));

ui->tableWidget->setItem(9, 1, new QTableWidgetItem(""));

ui->tableWidget->setItem(10, 1, new QTableWidgetItem(""));

ui->tableWidget->setItem(11, 1, new QTableWidgetItem(""));

ui->tableWidget->setItem(12, 1, new QTableWidgetItem(""));

ui->tableWidget->setItem(13, 1, new QTableWidgetItem(""));

}

//DayEXPInfo

if(now\_index[0] != -1 && now\_index[1] != -1 && now\_index[2] != -1)

{

p\_Route p1=hRoute;

for(int i = 0; i < now\_index[2]; ++i)//find position

{

p1 = p1->nextRoute;

}

p\_DayInfo p2 = p1->hDayInfo;

for(int i = 0; i < now\_index[1]; ++i)

{

p2 = p2->nextDayInfo;

}

p\_DayEXPInfo p3=p2->hDayEXPInfo;

for(int i = 0; i < now\_index[0]; ++i)

{

p3 = p3->nextDayEXPInfo;

}

if(p3==NULL)

{

return;

}

ui->tableWidget->setItem(0, 0, new QTableWidgetItem("所属行程编号"));//show information

ui->tableWidget->setItem(1, 0, new QTableWidgetItem("序号"));

ui->tableWidget->setItem(2, 0, new QTableWidgetItem("顺序号"));

ui->tableWidget->setItem(3, 0, new QTableWidgetItem("交易时间"));

ui->tableWidget->setItem(4, 0, new QTableWidgetItem("交易种类"));

ui->tableWidget->setItem(5, 0, new QTableWidgetItem("交易金额"));

ui->tableWidget->setItem(6, 0, new QTableWidgetItem("是否预支付"));

ui->tableWidget->setItem(7, 0, new QTableWidgetItem("支付方式"));

ui->tableWidget->setItem(8, 0, new QTableWidgetItem("说明"));

ui->tableWidget->setItem(9, 0, new QTableWidgetItem(""));

ui->tableWidget->setItem(10, 0, new QTableWidgetItem(""));

ui->tableWidget->setItem(11, 0, new QTableWidgetItem(""));

ui->tableWidget->setItem(12, 0, new QTableWidgetItem(""));

ui->tableWidget->setItem(13, 0, new QTableWidgetItem(""));

ui->tableWidget->setItem(0, 1, new QTableWidgetItem(QString("%1").arg(p3->routeNum)));

ui->tableWidget->setItem(1, 1, new QTableWidgetItem(QString("%1").arg(p2->order)));

ui->tableWidget->setItem(2, 1, new QTableWidgetItem(QString("%1").arg(p3->serialNum)));

ui->tableWidget->setItem(3, 1, new QTableWidgetItem(QString::fromUtf8(p3->TxTime)));

ui->tableWidget->setItem(4, 1, new QTableWidgetItem(QString::fromUtf8(p3->TxKind)));

ui->tableWidget->setItem(5, 1, new QTableWidgetItem(QString("%1").arg(p3->TxAmount)));

ui->tableWidget->setItem(6, 1, new QTableWidgetItem(QString::fromUtf8(p3->isPrepayment)));

ui->tableWidget->setItem(7, 1, new QTableWidgetItem(QString::fromUtf8(p3->payment)));

ui->tableWidget->setItem(8, 1, new QTableWidgetItem(QString::fromUtf8(p3->explain)));

ui->tableWidget->setItem(9, 1, new QTableWidgetItem(""));

ui->tableWidget->setItem(10, 1, new QTableWidgetItem(""));

ui->tableWidget->setItem(11, 1, new QTableWidgetItem(""));

ui->tableWidget->setItem(12, 1, new QTableWidgetItem(""));

ui->tableWidget->setItem(13, 1, new QTableWidgetItem(""));

}

}

MainWindow::~*MainWindow*()

{

delete ui;

}

void MainWindow::newRouteSlot()

{

newRoute\* dialog = new newRoute;

dialog->show();//unmodal dialog

dialog->*exec*();//modal dialog

showTree();

}

void MainWindow::newDayInfoSlot()

{

newDayInfo\* dailog = new newDayInfo;

dailog->*exec*();//modal dialog

showTree();

}

void MainWindow::newDayEXPInfoSlot()

{

newDayEXPInfo\* dialog = new newDayEXPInfo;

//dialog->show();//unmodal dialog

dialog->*exec*();//modal dialog

showTree();

}

void MainWindow::showAllslot()

{

showTree();

p\_Route p1 = hRoute;

QMessageBox msgBox;

if (p1 == NULL)

{

msgBox.setText("当前无信息 输出错误");

msgBox.*exec*();

return;

}

QString str;

while (p1 != NULL)

{

//printf("%d\n",p1->routeNum);

str = str + QString("%1").arg(p1->routeNum) + "\n";

p\_DayInfo p2 = p1->hDayInfo;

while (p2 != NULL)

{

//printf("\t%d\n", p2->order);

str = str + "\t" + QString("%1").arg(p2->order) + "\n";

p\_DayEXPInfo p3 = p2->hDayEXPInfo;

while (p3 != NULL)

{

//printf("\t\t%d\n", p3->serialNum);

str = str + "\t\t" + QString("%1").arg(p3->serialNum) + "\n";

p3 = p3->nextDayEXPInfo;

}

p2 = p2->nextDayInfo;

}

p1 = p1->nextRoute;

}

msgBox.setText(str);

msgBox.*exec*();

return;

}

void MainWindow::deleteRouteSlot()

{

deletedialog = new QDialog;

deletedialog->setWindowTitle("删除行程");

QGroupBox \*box = new QGroupBox(this);

QPushButton \*submitBtn = new QPushButton("确认");

QPushButton \*cancelBtn = new QPushButton("取消");

connect(submitBtn, SIGNAL(clicked(bool)), this, SLOT(Act\_deleteRouteSlot()));

connect(cancelBtn, SIGNAL(clicked(bool)), this, SLOT(cancelBtnSlot()));

QLabel \*routNum = new QLabel("行程编号");

routeNumLineEdit = new QLineEdit;

QGridLayout \*layout = new QGridLayout;

layout->addWidget(routNum, 0, 0, 1, 1); //从哪里开始 跨度

layout->addWidget(routeNumLineEdit, 0, 1, 1, 1); //从哪里开始 跨度

layout->addWidget(submitBtn, 1, 0, 1, 1);

layout->addWidget(cancelBtn, 1, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

deletedialog->setLayout(mainLayout);

deletedialog->show();

}

void MainWindow::Act\_deleteRouteSlot()

{

int n1 = routeNumLineEdit->text().toInt();

QMessageBox msgBox;

deletedialog->close();

if (deleteRoute(n1))

{

showTree();

msgBox.setText("删除成功");

msgBox.*exec*();

return;

}

else

{

msgBox.setText("删除失败");

msgBox.*exec*();

return;

}

}

void MainWindow::deleteDayInfoSlot()

{

deletedialog = new QDialog;

deletedialog->setWindowTitle("删除每日信息");

QGroupBox \*box = new QGroupBox(this);

QPushButton \*submitBtn = new QPushButton("确认");

QPushButton \*cancelBtn = new QPushButton("取消");

connect(submitBtn, SIGNAL(clicked(bool)), this, SLOT(Act\_deleteDayInfoSlot()));

connect(cancelBtn, SIGNAL(clicked(bool)), this, SLOT(cancelBtnSlot()));

QLabel \*routNum = new QLabel("行程编号");

routeNumLineEdit = new QLineEdit;

QLabel \*order = new QLabel("序号");

orderLineEdit = new QLineEdit;

QGridLayout \*layout = new QGridLayout;

layout->addWidget(routNum, 0, 0, 1, 1); //从哪里开始 跨度

layout->addWidget(routeNumLineEdit, 0, 1, 1, 1); //从哪里开始 跨度

layout->addWidget(order, 1, 0, 1, 1); //从哪里开始 跨度

layout->addWidget(orderLineEdit, 1, 1, 1, 1); //从哪里开始 跨度

layout->addWidget(submitBtn, 2, 0, 1, 1);

layout->addWidget(cancelBtn, 2, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

deletedialog->setLayout(mainLayout);

deletedialog->show();

}

void MainWindow::Act\_deleteDayInfoSlot()

{

QMessageBox msgBox;

int n1 = routeNumLineEdit->text().toInt();

int n2 = orderLineEdit->text().toInt();

deletedialog->close();

if (deleteDayInfo(n1, n2))

{

showTree();

msgBox.setText("删除成功");

msgBox.*exec*();

return;

}

else

{

msgBox.setText("删除失败");

msgBox.*exec*();

return;

}

}

void MainWindow::deleteDayEXPInfoSlot()

{

deletedialog = new QDialog;

deletedialog->setWindowTitle("删除每日信息");

QGroupBox \*box = new QGroupBox(this);

QPushButton \*submitBtn = new QPushButton("确认");

QPushButton \*cancelBtn = new QPushButton("取消");

connect(submitBtn, SIGNAL(clicked(bool)), this, SLOT(Act\_deleteDayEXPInfoSlot()));

connect(cancelBtn, SIGNAL(clicked(bool)), this, SLOT(cancelBtnSlot()));

QLabel \*routNum = new QLabel("行程编号");

routeNumLineEdit = new QLineEdit;

QLabel \*order = new QLabel("序号");

orderLineEdit = new QLineEdit;

QLabel \*serial = new QLabel("");

serialNumLineEdit = new QLineEdit;

QGridLayout \*layout = new QGridLayout;

layout->addWidget(routNum, 0, 0, 1, 1);

layout->addWidget(routeNumLineEdit, 0, 1, 1, 1);

layout->addWidget(order, 1, 0, 1, 1); //从哪里开始 跨度

layout->addWidget(orderLineEdit, 1, 1, 1, 1);

layout->addWidget(serial, 2, 0, 1, 1);

layout->addWidget(serialNumLineEdit, 2, 1, 1, 1);

layout->addWidget(submitBtn, 3, 0, 1, 1);

layout->addWidget(cancelBtn, 3, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

deletedialog->setLayout(mainLayout);

deletedialog->show();

}

void MainWindow::Act\_deleteDayEXPInfoSlot()

{

QMessageBox msgBox;

int n1 = routeNumLineEdit->text().toInt();

int n2 = orderLineEdit->text().toInt();

int n3 = serialNumLineEdit->text().toInt();

deletedialog->close();

if (deleteDayEXPInfo(n1, n2, n3))

{showTree();

showTree();

msgBox.setText("删除成功");

msgBox.*exec*();

return;

}

else

{

msgBox.setText("删除失败");

msgBox.*exec*();

return;

}

}

void MainWindow::quitBtnSLot()

{

deletedialog->close();

}

void MainWindow::showRouteslot()

{

showTree();

deletedialog = new QDialog;

deletedialog->setWindowTitle("查询行程");

QGroupBox \*box = new QGroupBox(this);

QPushButton \*submitBtn = new QPushButton("确认");

QPushButton \*cancelBtn = new QPushButton("取消");

connect(submitBtn, SIGNAL(clicked(bool)), this, SLOT(Act\_showRouteslot()));

connect(cancelBtn, SIGNAL(clicked(bool)), this, SLOT(cancelBtSlot()));

QLabel \*routNum = new QLabel("行程编号");

routeNumLineEdit = new QLineEdit;

QGridLayout \*layout = new QGridLayout;

layout->addWidget(routNum, 0, 0, 1, 1); //从哪里开始 跨度

layout->addWidget(routeNumLineEdit, 0, 1, 1, 1); //从哪里开始 跨度

layout->addWidget(submitBtn, 1, 0, 1, 1);

layout->addWidget(cancelBtn, 1, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

deletedialog->setLayout(mainLayout);

deletedialog->show();

}

void MainWindow::Act\_showRouteslot()

{

int n1 = routeNumLineEdit->text().toInt();

//查询信息并显示

QMessageBox msgBox;

p\_Route p1 = findRouteByNum(n1);

if (p1 == NULL)

{

deletedialog->close();

msgBox.setText("NOT FOUND");

msgBox.*exec*();

return;

}

deletedialog->close();

showDialog = new QDialog;

showDialog->setWindowTitle("行程信息");

QGroupBox \*box = new QGroupBox(this);

//浩大的工程啊

QLabel \*routeNum = new QLabel(QObject::tr("行程编号"));

QLabel \*s\_routeNum = new QLabel();//s代表show

QLabel \*routeName = new QLabel("主要目的地名称");

QLabel \*s\_routeName = new QLabel();

QLabel \*routeCountry = new QLabel("目的地所属国家/省份");

QLabel \*s\_routeCountry = new QLabel();

QLabel \*minT = new QLabel("最低温度");

QLabel \*s\_minT = new QLabel();

QLabel \*maxT = new QLabel("最高温度");

QLabel \*s\_maxT = new QLabel();

QLabel \*recommendedDressing = new QLabel("推荐的着装");

QLabel \*s\_recommendedDressing = new QLabel();

QLabel \*bigTrans = new QLabel("大交通");

QLabel \*s\_bigTrans = new QLabel();

QLabel \*mainScene = new QLabel("主要景点");

QLabel \*s\_mainScene = new QLabel();

QLabel \*remoteNum = new QLabel("异地办理手续行程编号");

QLabel \*s\_remoteNum = new QLabel();

QLabel \*beginDate = new QLabel("出发日期");

QLabel \*s\_beginDate = new QLabel();

QLabel \*arriveDate = new QLabel("返回到达日期");

QLabel \*s\_arriveDate = new QLabel();

QLabel \*totalTime = new QLabel("行程总天数");

QLabel \*s\_totalTime = new QLabel();

QLabel \*totalNum = new QLabel("行程总人数");

QLabel \*s\_totalNum = new QLabel();

QLabel \*totalPrice = new QLabel("行程总费用");

QLabel \*s\_totalPrice = new QLabel();

//体力活

//str1=QString("%1").arg(p1->routeNum);

s\_routeNum->setText(QString("%1").arg(p1->routeNum));

s\_routeName->setText(QString::fromUtf8(p1->routeName));

s\_routeCountry->setText(QString::fromUtf8(p1->routeCountry));

s\_minT->setText(QString("%1").arg(p1->dressingIndex.minT));

s\_maxT->setText(QString("%1").arg(p1->dressingIndex.maxT));

s\_recommendedDressing->setText(QString::fromUtf8(p1->dressingIndex.recommendedDressing));

s\_bigTrans->setText(QString::fromUtf8(p1->bigTrans));

s\_mainScene->setText(QString::fromUtf8(p1->mainScene));

s\_remoteNum->setText(QString::fromUtf8(p1->remoteNum));

s\_beginDate->setText(QString::fromUtf8(p1->beginDate));

s\_arriveDate->setText(QString::fromUtf8(p1->arriveDate));

s\_totalTime->setText(QString("%1").arg(p1->totalTime));

s\_totalNum->setText(QString("%1").arg(p1->totalNum));

s\_totalPrice->setText(QString("%1").arg(p1->totalPrice));

//QLineEdit \*route\_routeNumLineEdit = new QLineEdit();//注意声明形式

QGridLayout \*layout = new QGridLayout;

layout->addWidget(routeNum, 0, 0, 1, 1);

layout->addWidget(s\_routeNum, 0, 1, 1, 1);

layout->addWidget(routeName, 1, 0, 1, 1);

layout->addWidget(s\_routeName, 1, 1, 1, 1);

layout->addWidget(routeCountry, 2, 0, 1, 1);

layout->addWidget(s\_routeCountry, 2, 1, 1, 1);

layout->addWidget(minT, 3, 0, 1, 1);

layout->addWidget(s\_minT, 3, 1, 1, 1);

layout->addWidget(maxT, 4, 0, 1, 1);

layout->addWidget(s\_maxT, 4, 1, 1, 1);

layout->addWidget(recommendedDressing, 5, 0, 1, 1);

layout->addWidget(s\_recommendedDressing, 5, 1, 1, 1);

layout->addWidget(bigTrans, 6, 0, 1, 1);

layout->addWidget(s\_bigTrans, 6, 1, 1, 1);

layout->addWidget(mainScene, 7, 0, 1, 1);

layout->addWidget(s\_mainScene, 7, 1, 1, 1);

layout->addWidget(remoteNum, 8, 0, 1, 1);

layout->addWidget(s\_remoteNum, 8, 1, 1, 1);

layout->addWidget(beginDate, 9, 0, 1, 1);

layout->addWidget(s\_beginDate, 9, 1, 1, 1);

layout->addWidget(arriveDate, 10, 0, 1, 1);

layout->addWidget(s\_arriveDate, 10, 1, 1, 1);

layout->addWidget(totalTime, 11, 0, 1, 1);

layout->addWidget(s\_totalTime, 11, 1, 1, 1);

layout->addWidget(totalNum, 12, 0, 1, 1);

layout->addWidget(s\_totalNum, 12, 1, 1, 1);

layout->addWidget(totalPrice, 13,0, 1, 1);

layout->addWidget(s\_totalPrice, 13, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

showDialog->setLayout(mainLayout);

showDialog->show();

}

void MainWindow::showDayInfoslot()

{

deletedialog = new QDialog;

deletedialog->setWindowTitle("显示每日信息");

QGroupBox \*box = new QGroupBox(this);

QPushButton \*submitBtn = new QPushButton("确认");

QPushButton \*cancelBtn = new QPushButton("取消");

connect(submitBtn, SIGNAL(clicked(bool)), this, SLOT(Act\_showDayInfoSlot()));

connect(cancelBtn, SIGNAL(clicked(bool)), this, SLOT(cancelBtnSlot()));

QLabel \*routNum = new QLabel("行程编号");

routeNumLineEdit = new QLineEdit;

QLabel \*order = new QLabel("序号");

orderLineEdit = new QLineEdit;

QGridLayout \*layout = new QGridLayout;

layout->addWidget(routNum, 0, 0, 1, 1); //从哪里开始 跨度

layout->addWidget(routeNumLineEdit, 0, 1, 1, 1); //从哪里开始 跨度

layout->addWidget(order, 1, 0, 1, 1); //从哪里开始 跨度

layout->addWidget(orderLineEdit, 1, 1, 1, 1); //从哪里开始 跨度

layout->addWidget(submitBtn, 2, 0, 1, 1);

layout->addWidget(cancelBtn, 2, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

deletedialog->setLayout(mainLayout);

deletedialog->show();

}

void MainWindow::Act\_showDayInfoSlot()

{

QMessageBox msgBox;

int n1 = routeNumLineEdit->text().toInt();

int n2 = orderLineEdit->text().toInt();

p\_DayInfo p2=findDayInfoByOrder(n1,n2);

if (p2 == NULL)

{

deletedialog->close();

msgBox.setText("NOT FOUND");

msgBox.*exec*();

return;

}

deletedialog->close();

showDialog = new QDialog;

showDialog->setWindowTitle("行程信息");

QGroupBox \*box = new QGroupBox(this);

//浩大的工程啊

QLabel \*routeNum = new QLabel(QObject::tr("所属行程编号"));

QLabel \*s\_routeNum = new QLabel();//s代表show

QLabel \*order = new QLabel(QObject::tr("序号"));

QLabel \*s\_order = new QLabel();//s代表show

QLabel \*breakfastLoc = new QLabel("早餐地点");

QLabel \*s\_breakfastLoc = new QLabel();

QLabel \*morningScene = new QLabel("上午景点");

QLabel \*s\_morningScene = new QLabel();

QLabel \*LunchLoc = new QLabel("午餐地点");

QLabel \*s\_LunchLoc = new QLabel();

QLabel \*afternoonScene = new QLabel("下午景点");

QLabel \*s\_afternoonScene = new QLabel();

QLabel \*dinnerLoc = new QLabel("晚餐地点");

QLabel \*s\_dinnerLoc = new QLabel();

QLabel \*transInfo = new QLabel("小交通");

QLabel \*s\_transInfo = new QLabel();

QLabel \*residence = new QLabel("住宿地点");

QLabel \*s\_residence = new QLabel();

//str1=QString("%1").arg(p1->routeNum);

s\_routeNum->setText(QString("%1").arg(p2->routeNum));

s\_order->setText(QString("%1").arg(p2->order));

s\_breakfastLoc->setText(QString::fromUtf8(p2->breakfastLoc));

s\_morningScene->setText(QString::fromUtf8(p2->morningScene));

s\_LunchLoc->setText(QString::fromUtf8(p2->LunchLoc));

s\_afternoonScene->setText(QString::fromUtf8(p2->afternoonScene));

s\_dinnerLoc->setText(QString::fromUtf8(p2->dinnerLoc));

s\_transInfo->setText(QString::fromUtf8(p2->transInfo));

s\_residence->setText(QString::fromUtf8(p2->residence));

//QLineEdit \*route\_routeNumLineEdit = new QLineEdit();//注意声明形式

QGridLayout \*layout = new QGridLayout;

layout->addWidget(routeNum, 0, 0, 1, 1);

layout->addWidget(s\_routeNum, 0, 1, 1, 1);

layout->addWidget(order, 1, 0, 1, 1);

layout->addWidget(s\_order, 1, 1, 1, 1);

layout->addWidget(breakfastLoc, 2, 0, 1, 1);

layout->addWidget(s\_breakfastLoc, 2, 1, 1, 1);

layout->addWidget(morningScene, 3, 0, 1, 1);

layout->addWidget(s\_morningScene, 3, 1, 1, 1);

layout->addWidget(LunchLoc, 4, 0, 1, 1);

layout->addWidget(s\_LunchLoc, 4, 1, 1, 1);

layout->addWidget(afternoonScene, 5, 0, 1, 1);

layout->addWidget(s\_afternoonScene, 5, 1, 1, 1);

layout->addWidget(dinnerLoc, 6, 0, 1, 1);

layout->addWidget(s\_dinnerLoc, 6, 1, 1, 1);

layout->addWidget(transInfo, 7, 0, 1, 1);

layout->addWidget(s\_transInfo, 7, 1, 1, 1);

layout->addWidget(residence, 8, 0, 1, 1);

layout->addWidget(s\_residence, 8, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

showDialog->setLayout(mainLayout);

showDialog->show();

}

void MainWindow::showDayEXPInfoslot()

{

deletedialog = new QDialog;

deletedialog->setWindowTitle("删除每日信息");

QGroupBox \*box = new QGroupBox(this);

QPushButton \*submitBtn = new QPushButton("确认");

QPushButton \*cancelBtn = new QPushButton("取消");

connect(submitBtn, SIGNAL(clicked(bool)), this, SLOT(Act\_showDayEXPInfoSlot()));

connect(cancelBtn, SIGNAL(clicked(bool)), this, SLOT(cancelBtnSlot()));

QLabel \*routNum = new QLabel("行程编号");

routeNumLineEdit = new QLineEdit;

QLabel \*order = new QLabel("序号");

orderLineEdit = new QLineEdit;

QLabel \*serial = new QLabel("序列号");

serialNumLineEdit = new QLineEdit;

QGridLayout \*layout = new QGridLayout;

layout->addWidget(routNum, 0, 0, 1, 1);

layout->addWidget(routeNumLineEdit, 0, 1, 1, 1);

layout->addWidget(order, 1, 0, 1, 1); //从哪里开始 跨度

layout->addWidget(orderLineEdit, 1, 1, 1, 1);

layout->addWidget(serial, 2, 0, 1, 1);

layout->addWidget(serialNumLineEdit, 2, 1, 1, 1);

layout->addWidget(submitBtn, 3, 0, 1, 1);

layout->addWidget(cancelBtn, 3, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

deletedialog->setLayout(mainLayout);

deletedialog->show();

}

void MainWindow::Act\_showDayEXPInfoSlot()

{

QMessageBox msgBox;

int n1 = routeNumLineEdit->text().toInt();

int n2 = orderLineEdit->text().toInt();

int n3 = serialNumLineEdit->text().toInt();

deletedialog->close();

p\_DayEXPInfo p3=findDayEXPInfoByserialNum(n1,n2,n3);

if (p3==NULL)

{

msgBox.setText("NOT FOUND");

msgBox.*exec*();

return;

}

// (); //此处有问题

deletedialog->close();

showDialog = new QDialog;

showDialog->setWindowTitle("消费信息");

QGroupBox \*box = new QGroupBox(this);

//浩大的工程啊

QLabel \*routeNum = new QLabel(QObject::tr("所属行程编号"));

QLabel \*s\_routeNum = new QLabel();//s代表show

QLabel \*order = new QLabel(QObject::tr("序号"));

QLabel \*s\_order = new QLabel();//s代表show

QLabel \*serialNum = new QLabel("顺序号");

QLabel \*s\_serialNum = new QLabel();

QLabel \*TxTime = new QLabel("交易时间");

QLabel \*s\_TxTime = new QLabel();

QLabel \*TxKind = new QLabel("交易种类");

QLabel \*s\_TxKind = new QLabel();

QLabel \*TxAmount = new QLabel("交易金额");

QLabel \*s\_TxAmount = new QLabel();

QLabel \*isPrepayment = new QLabel("是否预支付");

QLabel \*s\_isPrepayment = new QLabel();

QLabel \*payment = new QLabel("支付方式");

QLabel \*s\_payment = new QLabel();

QLabel \*explain = new QLabel("住宿地点");

QLabel \*s\_explain = new QLabel();

//str1=QString("%1").arg(p1->routeNum);

s\_routeNum->setText(QString("%1").arg(p3->routeNum));

s\_order->setText(QString("%1").arg(p3->order));

s\_serialNum->setText(QString("%1").arg(p3->serialNum));

s\_TxTime->setText(QString::fromUtf8(p3->TxTime));

s\_TxKind->setText(QString::fromUtf8(p3->TxKind));

s\_TxAmount->setText(QString("%1").arg(p3->TxAmount));

s\_isPrepayment->setText(QString::fromUtf8(p3->isPrepayment));

s\_payment->setText(QString::fromUtf8(p3->payment));

s\_explain->setText(QString::fromUtf8(p3->explain));

//QLineEdit \*route\_routeNumLineEdit = new QLineEdit();//注意声明形式

QGridLayout \*layout = new QGridLayout;

layout->addWidget(routeNum, 0, 0, 1, 1);

layout->addWidget(s\_routeNum, 0, 1, 1, 1);

layout->addWidget(order, 1, 0, 1, 1);

layout->addWidget(s\_order, 1, 1, 1, 1);

layout->addWidget(serialNum, 2, 0, 1, 1);

layout->addWidget(s\_serialNum, 2, 1, 1, 1);

layout->addWidget(TxTime, 3, 0, 1, 1);

layout->addWidget(s\_TxTime, 3, 1, 1, 1);

layout->addWidget(TxKind, 4, 0, 1, 1);

layout->addWidget(s\_TxKind, 4, 1, 1, 1);

layout->addWidget(TxAmount, 5, 0, 1, 1);

layout->addWidget(s\_TxAmount, 5, 1, 1, 1);

layout->addWidget(isPrepayment, 6, 0, 1, 1);

layout->addWidget(s\_isPrepayment, 6, 1, 1, 1);

layout->addWidget(payment, 7, 0, 1, 1);

layout->addWidget(s\_payment, 7, 1, 1, 1);

layout->addWidget(explain, 8, 0, 1, 1);

layout->addWidget(s\_explain, 8, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

showDialog->setLayout(mainLayout);

showDialog->show();

}

void MainWindow::saveToFileSlot()

{

QMessageBox msgBox;

if(saveToFile())

{

msgBox.setText("Save To File Successfully");

msgBox.*exec*();

return;

}

else

{

msgBox.setText("Save To File Fail");

msgBox.*exec*();

return;

}

}

void MainWindow::loadFromFileSlot()//注意第一次保存

{

QMessageBox msgBox;

if(loadFromFile())

{

showTree();

msgBox.setText("load From File Successfully");

msgBox.*exec*();

return;

}

else

{

showTree();

msgBox.setText("load From File fail");

msgBox.*exec*();

return;

}

}

void MainWindow::showTree()

{

ui->treeWidget->clear();//first clear old tree

QTreeWidgetItem \*All = new QTreeWidgetItem(ui->treeWidget,QStringList(QString("All")));

p\_Route p1 = hRoute;

if (p1 == NULL)

{

return;

}

QString str;

while (p1 != NULL)

{

QTreeWidgetItem \*routeItem = new QTreeWidgetItem(QStringList("Route"+QString::number(p1->routeNum, 10)));

All->addChild(routeItem);

p\_DayInfo p2 = p1->hDayInfo;

while (p2 != NULL)

{

QTreeWidgetItem \*dayInfoItem = new QTreeWidgetItem(QStringList("DayInfo"+QString::number(p2->order, 10)));

routeItem->addChild(dayInfoItem);

p\_DayEXPInfo p3 = p2->hDayEXPInfo;

while (p3 != NULL)

{

str = str + "\t\t" + QString("%1").arg(p3->serialNum) + "\n";

QTreeWidgetItem \*dayEXPInfoItem = new QTreeWidgetItem(QStringList("DayEXPInfo"+QString::number(p3->serialNum, 10)));

dayInfoItem->addChild(dayEXPInfoItem);

p3 = p3->nextDayEXPInfo;

}

p2 = p2->nextDayInfo;

}

p1 = p1->nextRoute;

}

ui->treeWidget->expandAll();//expend all nodes

return;

}

void MainWindow::on\_modifyButton\_clicked()

{

//Mouse\_position(item);//get mouse position

if(now\_index[0] == -1 && now\_index[1] == -1 && now\_index[2] == -1)//未点击

{

return;

}

//Route

if(now\_index[0] != -1 && now\_index[1] == -1 && now\_index[2] == -1)

{

p\_Route p1=hRoute;

//struct college\_info\* college\_temp = head;

for(int i = 0; i < now\_index[0]; ++i)

{//find position

p1 = p1->nextRoute;

}

if(p1 == NULL)

{

return ;

}

p1->routeNum=ui->tableWidget->item(0,1)->text().toInt();//int

strcpy(p1->routeName, ui->tableWidget->item(1,1)->text().toStdString().c\_str());

strcpy(p1->routeCountry, ui->tableWidget->item(2,1)->text().toStdString().c\_str());

p1->dressingIndex.minT=ui->tableWidget->item(3,1)->text().toInt();

p1->dressingIndex.maxT=ui->tableWidget->item(4,1)->text().toInt();

strcpy(p1->dressingIndex.recommendedDressing, ui->tableWidget->item(5,1)->text().toStdString().c\_str());

strcpy(p1->bigTrans, ui->tableWidget->item(6,1)->text().toStdString().c\_str());

strcpy(p1->mainScene, ui->tableWidget->item(7,1)->text().toStdString().c\_str());

strcpy(p1->remoteNum, ui->tableWidget->item(8,1)->text().toStdString().c\_str());

strcpy(p1->beginDate, ui->tableWidget->item(9,1)->text().toStdString().c\_str());

strcpy(p1->arriveDate, ui->tableWidget->item(10,1)->text().toStdString().c\_str());

p1->totalTime=ui->tableWidget->item(11,1)->text().toInt();//int 这个也是输入的

p1->totalNum=ui->tableWidget->item(12,1)->text().toInt();//int

p1->totalPrice=ui->tableWidget->item(13,1)->text().toFloat();//float

}

//DayInfo

if(now\_index[0] != -1 && now\_index[1] != -1 && now\_index[2] == -1)

{

p\_Route p1=hRoute;

for(int i = 0; i < now\_index[1]; ++i)//find position

{

p1 = p1->nextRoute;

}

p\_DayInfo p2 = p1->hDayInfo;

for(int i = 0; i < now\_index[0]; ++i)

{

p2 = p2->nextDayInfo;

}

if(p2 == NULL)

{

return ;

}

p2->routeNum=ui->tableWidget->item(0,1)->text().toInt();//int

p2->order=ui->tableWidget->item(1,1)->text().toInt();//int

strcpy(p2->breakfastLoc, ui->tableWidget->item(2,1)->text().toStdString().c\_str());

strcpy(p2->morningScene, ui->tableWidget->item(3,1)->text().toStdString().c\_str());

strcpy(p2->LunchLoc, ui->tableWidget->item(4,1)->text().toStdString().c\_str());

strcpy(p2->afternoonScene, ui->tableWidget->item(5,1)->text().toStdString().c\_str());

strcpy(p2->dinnerLoc, ui->tableWidget->item(6,1)->text().toStdString().c\_str());

strcpy(p2->transInfo, ui->tableWidget->item(7,1)->text().toStdString().c\_str());

strcpy(p2->residence, ui->tableWidget->item(8,1)->text().toStdString().c\_str());

}

//DayEXPInfo

if(now\_index[0] != -1 && now\_index[1] != -1 && now\_index[2] != -1)

{

p\_Route p1=hRoute;

for(int i = 0; i < now\_index[2]; ++i)//find position

{

p1 = p1->nextRoute;

}

p\_DayInfo p2 = p1->hDayInfo;

for(int i = 0; i < now\_index[1]; ++i)

{

p2 = p2->nextDayInfo;

}

p\_DayEXPInfo p3=p2->hDayEXPInfo;

for(int i = 0; i < now\_index[0]; ++i)

{

p3 = p3->nextDayEXPInfo;

}

if(p3==NULL)

{

return;

}

p3->routeNum=ui->tableWidget->item(0,1)->text().toInt();//int

p3->order=ui->tableWidget->item(1,1)->text().toInt();//int

p3->serialNum=ui->tableWidget->item(2,1)->text().toInt();//int

strcpy(p3->TxTime, ui->tableWidget->item(3,1)->text().toStdString().c\_str());

strcpy(p3->TxKind, ui->tableWidget->item(4,1)->text().toStdString().c\_str());

p3->TxAmount=ui->tableWidget->item(5,1)->text().toFloat();//float

strcpy(p3->isPrepayment, ui->tableWidget->item(6,1)->text().toStdString().c\_str());

strcpy(p3->payment, ui->tableWidget->item(7,1)->text().toStdString().c\_str());

strcpy(p3->explain, ui->tableWidget->item(8,1)->text().toStdString().c\_str());

}

showTree();

}

void MainWindow::countEXPSlot()

{

showTree();

QTableWidget \* tableWidget = new QTableWidget(this);

tableWidget->setRowCount(15);

tableWidget->setColumnCount(10);

tableWidget->setHorizontalHeaderLabels(QStringList() << "费用类型" <<"费用金额");

tableWidget->verticalHeader()->*setVisible*(false);//hide vertical header

tableWidget->setColumnWidth(0,200);

tableWidget->setColumnWidth(1,200);

float residence[10]={0};

float bigTrans[10]={0};

float littleTrans[10]={0};

float food[10]={0};

float tickets[10]={0};

float shopping[10]={0};

float others[10]={0};

float pp\_residence=0;

float pp\_bigTrans=0;

float pp\_litteTrans=0;

float pp\_food=0;

float pp\_tickets=0;

float pp\_shopping=0;

float pp\_others=0;

float total;

tableWidget->setItem(0, 0, new QTableWidgetItem("行程编号："));//show information

tableWidget->setItem(1, 0, new QTableWidgetItem("住宿"));//show information

tableWidget->setItem(2, 0, new QTableWidgetItem("大交通"));

tableWidget->setItem(3, 0, new QTableWidgetItem("小交通"));

tableWidget->setItem(4, 0, new QTableWidgetItem("餐饮"));

tableWidget->setItem(5, 0, new QTableWidgetItem("门票"));

tableWidget->setItem(6, 0, new QTableWidgetItem("购物"));

tableWidget->setItem(7, 0, new QTableWidgetItem("其他"));

tableWidget->setItem(8, 0, new QTableWidgetItem("总计"));

//QPieSlice \*slice\_red = series->slices().at(0);

//QPieSlice \*slice\_green = series->slices().at(1);

//QPieSlice \*slice\_blue = series->slices().at(2);

//slice\_red->setColor(QColor(255,0,0,255));

//slice\_green->setColor(QColor(0,255,0,255));

//slice\_blue->setColor(QColor(0,0,255,255));

QChart \*chart[10];//最多显示10个图表

QPieSeries \*series[10];

QChartView \*chartview[10];

showDialog = new QDialog;

showDialog->setWindowTitle("行程信息");

QGroupBox \*box = new QGroupBox(this);

QGridLayout \*layout = new QGridLayout;

box->setLayout(layout);

layout->addWidget(tableWidget,0,0);

p\_Route p1 = hRoute;

int t1=1;//最终数目为t1-1

int t2=0;

int t3=0;

if (p1 == NULL)

{

//MessageBox msgBox;

// msgBox.setText("当前无信息 输出错误");

//msgBox.exec();

return;

}

while (p1 != NULL)

{

//str = str + QString("%1").arg(p1->routeNum) + "\n";

p\_DayInfo p2 = p1->hDayInfo;

while (p2 != NULL)

{

//str = str + "\t" + QString("%1").arg(p2->order) + "\n";

p\_DayEXPInfo p3 = p2->hDayEXPInfo;

while (p3 != NULL)

{

//str = str + "\t\t" + QString("%1").arg(p3->serialNum) + "\n";

if(strcmp(p3->TxKind, "住宿")==0)

{

residence[t1]+=p3->TxAmount;

}

else

{

if(strcmp(p3->TxKind, "大交通")==0)

{

bigTrans[t1]+=p3->TxAmount;

}

else

{

if(strcmp(p3->TxKind, "小交通")==0)

{

littleTrans[t1]+=p3->TxAmount;

}

else

{

if(strcmp(p3->TxKind, "餐饮")==0)

{

food[t1]+=p3->TxAmount;

}

else

{

if(strcmp(p3->TxKind, "门票")==0)

{

tickets[t1]+=p3->TxAmount;

}

else

{

if(strcmp(p3->TxKind, "购物")==0)

{

shopping[t1]+=p3->TxAmount;

}

else

{

others[t1]+=p3->TxAmount;

}

}

}

}

}

}

p3=p3->nextDayEXPInfo;

t3++;

}

p2 = p2->nextDayInfo;

t2++;

}

p1 = p1->nextRoute;//为什么不显示？

t1++;

}

p1=hRoute;

int index\_mostTickets=1;

for(int i=1;i<t1;i++)

{

chart[i]=new QChart();

series[i]= new QPieSeries();

tableWidget->setItem(0, i, new QTableWidgetItem(QString("%1").arg(p1->routeNum)));

tableWidget->setItem(1, i, new QTableWidgetItem(QString("%1").arg(residence[i])));

tableWidget->setItem(2, i, new QTableWidgetItem(QString("%1").arg(bigTrans[i])));

tableWidget->setItem(3, i, new QTableWidgetItem(QString("%1").arg(littleTrans[i])));

tableWidget->setItem(4, i, new QTableWidgetItem(QString("%1").arg(food[i])));

tableWidget->setItem(5, i, new QTableWidgetItem(QString("%1").arg(tickets[i])));

tableWidget->setItem(6, i, new QTableWidgetItem(QString("%1").arg(shopping[i])));

tableWidget->setItem(7, i, new QTableWidgetItem(QString("%1").arg(others[i])));

total=residence[i]+bigTrans[i]+littleTrans[i]+food[i]+tickets[i]+shopping[i]+others[i];

tableWidget->setItem(8, i, new QTableWidgetItem(QString("%1").arg(total)));

pp\_residence=residence[i]/total\*100;

pp\_bigTrans=bigTrans[i]/total\*100;

pp\_litteTrans=littleTrans[i]/total\*100;

pp\_food=food[i]/total\*100;

pp\_tickets=tickets[i]/total\*100;

pp\_shopping=shopping[i]/total\*100;

pp\_others=others[i]/total\*100;

series[i]->append("住宿"+QString("%1").arg(pp\_residence)+"%",residence[i]);

series[i]->append("大交通"+QString("%1").arg(pp\_bigTrans)+"%",bigTrans[i]);

series[i]->append("小交通"+QString("%1").arg(pp\_litteTrans)+"%",littleTrans[i]);

series[i]->append("餐饮"+QString("%1").arg(pp\_food)+"%",food[i]);

series[i]->append("门票"+QString("%1").arg(pp\_tickets)+"%",tickets[i]);

series[i]->append("购物"+QString("%1").arg(pp\_shopping)+"%",shopping[i]);

series[i]->append("其他"+QString("%1").arg(pp\_others)+"%",others[i]);

series[i]->setLabelsVisible();

chart[i]->legend()->setAlignment(Qt::AlignRight); // 设置图例靠右显示

chart[i]->addSeries(series[i]);

chart[i]->setTitle(QString("%1").arg(p1->routeNum)+"消费信息");

chart[i]->legend()->hide();

chartview[i]= new QChartView(chart[i]);

chartview[i]->setRenderHint(QPainter::Antialiasing);

layout->addWidget(chartview[i],0,i);

p1->totalPrice=total;

if(tickets[i]>tickets[index\_mostTickets])

{

index\_mostTickets=i;

}

p1=p1->nextRoute;

}

p1=hRoute;

int index\_mostAvg=1;

int index\_leastAvg=1;

float avg[10];

for(int i=1;i<t1;i++)

{

if(p1->totalTime!=0)

{

avg[i]=p1->totalPrice/p1->totalTime;

if(avg[i]>avg[index\_mostAvg])

{

index\_mostAvg=i;

}

if(avg[i]<avg[index\_leastAvg])

{

index\_leastAvg=i;

}

}

p1=p1->nextRoute;

}

tableWidget->setItem(9, 0, new QTableWidgetItem("门票价格最高的是"));

tableWidget->setItem(9, 1, new QTableWidgetItem(tableWidget->item(0,index\_mostTickets)->text()));

tableWidget->setItem(9, 2, new QTableWidgetItem(QString("%1").arg(tickets[index\_mostTickets])+"元"));

tableWidget->setItem(10, 0, new QTableWidgetItem("日消费价格最高的是"));

tableWidget->setItem(10, 1, new QTableWidgetItem(tableWidget->item(0,index\_mostAvg)->text()));

tableWidget->setItem(10, 2, new QTableWidgetItem(QString("%1").arg(avg[index\_mostAvg])+"元"));

tableWidget->setItem(10, 0, new QTableWidgetItem("日消费价格最低的是"));

tableWidget->setItem(10, 1, new QTableWidgetItem(tableWidget->item(0,index\_leastAvg)->text()));

tableWidget->setItem(10, 2, new QTableWidgetItem(QString("%1").arg(avg[index\_leastAvg])+"元"));

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

showDialog->setLayout(mainLayout);

showDialog->show();

//进行统计图表输出QThanzi

}

void MainWindow::showTransInfoSlot()

{

deletedialog = new QDialog;

deletedialog->setWindowTitle("查询小交通信息");

QGroupBox \*box = new QGroupBox(this);

QPushButton \*submitBtn = new QPushButton("确认");

QPushButton \*cancelBtn = new QPushButton("取消");

connect(submitBtn, SIGNAL(clicked(bool)), this, SLOT(Act\_showTransInfoSlot()));

connect(cancelBtn, SIGNAL(clicked(bool)), this, SLOT(cancelBtSlot()));

QLabel \*routNum = new QLabel("景点名称");

routeNumLineEdit = new QLineEdit;

QGridLayout \*layout = new QGridLayout;

layout->addWidget(routNum, 0, 0, 1, 1); //从哪里开始 跨度

layout->addWidget(routeNumLineEdit, 0, 1, 1, 1); //从哪里开始 跨度

layout->addWidget(submitBtn, 1, 0, 1, 1);

layout->addWidget(cancelBtn, 1, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

deletedialog->setLayout(mainLayout);

deletedialog->show();

}

void MainWindow::Act\_showTransInfoSlot()

{

QMessageBox msgBox;

char Scene[50];

strcpy(Scene, routeNumLineEdit->text().toStdString().c\_str());

deletedialog->close();

p\_DayInfo tp2=findTransInfoByScene(Scene);

if (tp2!=NULL)

{

msgBox.setText("小交通信息："+QString::fromUtf8(tp2->transInfo));

msgBox.*exec*();

return;

}

else

{

msgBox.setText("查找失败");

msgBox.*exec*();

return;

}

}

void MainWindow::showResidenceSlot()

{

deletedialog = new QDialog;

deletedialog->setWindowTitle("查询小交通信息");

QGroupBox \*box = new QGroupBox(this);

QPushButton \*submitBtn = new QPushButton("确认");

QPushButton \*cancelBtn = new QPushButton("取消");

connect(submitBtn, SIGNAL(clicked(bool)), this, SLOT(Act\_showResidenceSlot()));

connect(cancelBtn, SIGNAL(clicked(bool)), this, SLOT(cancelBtSlot()));

QLabel \*routNum = new QLabel("景点名称");

routeNumLineEdit = new QLineEdit;

QGridLayout \*layout = new QGridLayout;

layout->addWidget(routNum, 0, 0, 1, 1); //从哪里开始 跨度

layout->addWidget(routeNumLineEdit, 0, 1, 1, 1); //从哪里开始 跨度

layout->addWidget(submitBtn, 1, 0, 1, 1);

layout->addWidget(cancelBtn, 1, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

deletedialog->setLayout(mainLayout);

deletedialog->show();

}

void MainWindow::Act\_showResidenceSlot()

{

QMessageBox msgBox;

char Residence[50];

strcpy(Residence, routeNumLineEdit->text().toStdString().c\_str());

deletedialog->close();

float EXP=findEXPByResidence(Residence);

if (EXP!=-1)

{

EXP+=1;

msgBox.setText(QString::fromUtf8(Residence)+"价格："+QString("%1").arg(EXP));

msgBox.*exec*();

return;

}

else

{

msgBox.setText("无查找旅店");

msgBox.*exec*();

return;

}

}

void MainWindow::showDressingIndexSlot()

{

deletedialog = new QDialog;

deletedialog->setWindowTitle("穿衣指数");

QGroupBox \*box = new QGroupBox(this);

QPushButton \*submitBtn = new QPushButton("确认");

QPushButton \*cancelBtn = new QPushButton("取消");

connect(submitBtn, SIGNAL(clicked(bool)), this, SLOT(Act\_showDressingIndexSlot()));

connect(cancelBtn, SIGNAL(clicked(bool)), this, SLOT(cancelBtSlot()));

QLabel \*time1 = new QLabel("时间1");

QLabel \*time2 = new QLabel("时间2");

routeNumLineEdit = new QLineEdit;

orderLineEdit= new QLineEdit;

QGridLayout \*layout = new QGridLayout;

layout->addWidget(time1, 0, 0, 1, 1);

layout->addWidget(routeNumLineEdit, 0, 1, 1, 1);

layout->addWidget(time2, 1, 0, 1, 1);

layout->addWidget(orderLineEdit, 1, 1, 1, 1);

layout->addWidget(submitBtn, 2, 0, 1, 1);

layout->addWidget(cancelBtn, 2, 1, 1, 1);

box->setLayout(layout);

QGridLayout \*mainLayout = new QGridLayout;

mainLayout->addWidget(box);

deletedialog->setLayout(mainLayout);

deletedialog->show();

}

void MainWindow::Act\_showDressingIndexSlot()

{

QMessageBox msgBox;

QDateTime dateTime1 = QDateTime::fromString(routeNumLineEdit->text(), "yyyy-MM-dd");

QDateTime dateTime2 = QDateTime::fromString(orderLineEdit->text(), "yyyy-MM-dd");

QDateTime t\_time1;

QDateTime t\_time2;

deletedialog->close();

p\_Route p1 = hRoute;

if (p1 == NULL)

{

msgBox.setText("无符合要求的行程信息");

msgBox.*exec*();

return;

}

QString str1;

QString str2;

QString printStr;

p\_Route A\_p1[10]= {NULL};

int i=1;

while (p1 != NULL)

{

str1=QString::fromUtf8(p1->beginDate);

str2=QString::fromUtf8(p1->arriveDate);

if(str1.isEmpty()||str2.isEmpty())

{

p1 = p1->nextRoute;

}

t\_time1=QDateTime::fromString(str1, "yyyy-MM-dd");

t\_time2=QDateTime::fromString(str2, "yyyy-MM-dd");

if(dateTime1<=t\_time1&&t\_time2<=dateTime2&&t\_time1<=t\_time2)

{

//入p3

A\_p1[i++]=p1;

}

p1 = p1->nextRoute;

}

i=1;

if(A\_p1[1]==NULL)

{

msgBox.setText("无符合要求的行程信息");

msgBox.*exec*();

return;

}

while(A\_p1[i]!=NULL)

{

QString qq=QString::fromUtf8(A\_p1[i]->dressingIndex.recommendedDressing);

QString tStr="行程："+QString("%1").arg(A\_p1[i]->routeNum)+"\t"+QString::fromUtf8(A\_p1[i]->dressingIndex.recommendedDressing)+"\n";

printStr+=tStr;

i++;

}

msgBox.setText(printStr);

msgBox.*exec*();

return;

}

void MainWindow::sortRouteSlot()

{

//sortByRouteNum();

//ListBubbleSort(&hRoute);

sortRouteByNum(&hRoute);

showTree();

}

void MainWindow::sortDayInfoSlot()

{

p\_Route p1=hRoute;

if(p1==NULL)

{

return ;

}

while(p1!=NULL)

{

p\_DayInfo p2=p1->hDayInfo;

sortDayInfoByOrder(&p2);

p1->hDayInfo=p2;

p1=p1->nextRoute;

}

showTree();

}

void MainWindow::sortDayEXPInfoSlot()

{

p\_Route p1=hRoute;

if(p1==NULL)

{

return ;

}

while(p1!=NULL)

{

p\_DayInfo p2=p1->hDayInfo;

if(p2!=NULL)

{

p\_DayEXPInfo p3=p2->hDayEXPInfo;

sortDayEXPInfoBySerial(&p3);

p2->hDayEXPInfo=p3;

p2=p2->nextDayInfo;

}

p1=p1->nextRoute;

}

showTree();

}

### main.cpp

#include "mainwindow.h"

#include"newRoute.h"

#include"Personaltrip.h"

#include <QApplication>

#include<QTextCodec>

int main(int argc, char \*argv[])

{

//此处可执行调试

QTextCodec \*codec = QTextCodec::codecForName("UTF-8");

QTextCodec::setCodecForLocale(QTextCodec::codecForLocale());

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}