**华中科技大学计算机科学与技术学院**

**《C语言与程序设计》课程设计**

**实验报告**

**题目： 物流信息管理系统**

**专业： 物联网工程**

**班级： 物联网1601**

**学号： U201614890**

**姓名： 徐光磊**

**成绩：**

**指导教师： 刘芳**

**完成日期： 2017年 9月 1日**

目录

[一．系统功能结构模块图 4](#_Toc493242034)

[二．数据结构设计及其用法说明 5](#_Toc493242035)

[1.货物信息结构体： 5](#_Toc493242036)

[2.车辆信息结构体 5](#_Toc493242037)

[3.站点信息结构体 5](#_Toc493242038)

[4.路线信息结构体 6](#_Toc493242039)

[5.使用方法 7](#_Toc493242040)

[6.数据类型及范例 7](#_Toc493242041)

[三．程序结构 10](#_Toc493242042)

[1.IO操作流程图（以加载所有存档为例）： 10](#_Toc493242043)

[2.删除结点流程图（以删除路线结点为例）： 11](#_Toc493242044)

[4.修改站点信息流程： 13](#_Toc493242045)

[5.查询流程（以查询查询公里数最短的路线举例）： 14](#_Toc493242046)

[6.生成统计报表流程： 15](#_Toc493242047)

[四．各模块的功能 16](#_Toc493242048)

[1.三种信息界面： 16](#_Toc493242049)

[2.快速查询界面： 16](#_Toc493242050)

[3.查看详细信息： 16](#_Toc493242051)

[4.统计生成表格： 16](#_Toc493242052)

[5.自动更新存档： 16](#_Toc493242053)

[6.增删改操作： 16](#_Toc493242054)

[五．测试 17](#_Toc493242055)

[1.系统首页： 17](#_Toc493242056)

[2.路线信息界面： 17](#_Toc493242057)

[3.快速查询界面： 18](#_Toc493242058)

[4.统计报表： 18](#_Toc493242059)

[5.修改信息界面（站点）： 19](#_Toc493242060)

[6.增添信息界面（车辆）： 20](#_Toc493242061)

[7.删除信息界面（站点）： 20](#_Toc493242062)

[六．体会 21](#_Toc493242063)

[七．致谢 22](#_Toc493242064)

[八．参考文献 22](#_Toc493242065)

[附录：源代码清单 23](#_Toc493242066)

[ListTool.h 23](#_Toc493242067)

[DataStructure.h 36](#_Toc493242068)

[Main.c 41](#_Toc493242069)

**物流信息管理系统**

# 一．系统功能结构模块图

# 二．数据结构设计及其用法说明

1.货物信息结构体：

/\*\*

\* 货物的基本信息：\*Type : 装货、卸货的货物的种类

\* \*Volume : 货物所占容量，（单位立方米）

\*/

typedef struct Good {

char uploadType[10];

float upVolume;

char downloadType[10];

float downVolume;

} good;

2.车辆信息结构体

/\*\*

\* 配送车辆的基本信息： carID： 车辆牌照

\* routeID：执行配送路线编号

\* driverName： 司机姓名

\* driverTel： 司机移动电话

\* carID: 车辆编号

\* goodList： 运货清单

\*/

typedef struct Car {

char carID[9];

char routeID[7];

char driverName[9];

char driverTel[12];

good \* good;

struct Car \* next;

} car;

3.站点信息结构体

/\*\*

\* 经停站点详细信息: routeNums: 固定配送路线数量

\* routeIDArray: 固定配送路线编号的数组

\* siteSID: 站点序号

\* siteID: 站点编号

\* siteNames: 站点名称

\* d2Start: 与初始站点距离

\* d2Last: 与上一个站点距离

\* t2Last: 与上一个站点交通耗时

\* tWait: 停留耗时

\* next: 指向下一个站点的指针

\*/

typedef struct Site {

char routeID[7];

int siteSID;

char siteID[11];

char siteName[51];

float d2Start;

float d2Last;

float time2Last;

float waitTime;

char routeIDArray[50];

struct Car \* carHeadP;

struct Site \* next;

} site;

4.路线信息结构体

/\*\*

\* 配送路线详细信息: routeID: 固定配送路线编号

\* name: 固定配送路线编号名称

\* siteNums: 固定配送路线总站点数

\* miles: 固定配送路线总公里数

\* period: 全站点配送总耗时

\* startSite: 起始站点编号

\* endSite: 终止站点编号

\* adminName: 负责人姓名

\* tel: 负责人办公室电话

\* mTel: 负责人移动电话

\* email: 负责人电子邮箱

\* firstSite: 指向站点（即第一个站点）的指针

\* next: 指向下一个路线的指针

\*/

typedef struct Route {

char routeID[7];

char name[21];

short siteNums;

float miles;

float period;

char startSite[11];

char endSite[11];

char adminName[9];

char tel[9];

char mTel[12];

char email[51];

struct Site \* firstSite;//指向该路线的第一个站点，并形成路线链表

struct Route \* next;

} route;

5.使用方法

以上的结构体均可直接通过malloc函数开辟内存空间后直接赋值使用，创建、插入、修改、删除、释放空间等操作均可直接调用ListTool.h文件中的链表相关函数来进行。

6.数据类型及范例

1. 路线

|  |  |  |
| --- | --- | --- |
| **中文字段名** | **类型及长度** | **举例** |
| 固定配送路线编号 | char[6] | r00001 |
| 固定配送路线名称 | char[20] | 一号线 |
| 固定配送路线总站点数 | short | 3 |
| 固定配送路线总公里数 | float | 35.3 |
| 全站点配送总耗时 | Float(分钟) | 63.1 |
| 起始站点编号 | char[10] | s00001 |
| 终止站点编号 | char[10] | s00003 |
| 负责人姓名 | char[8] | 张三 |
| 负责人办公室电话 | char[8] | 89641687 |
| 负责人移动电话 | char[11] | 13521478664 |
| 负责人电子邮箱 | char[50] | ZhangThree@gmail.com |

1. 站点

|  |  |  |
| --- | --- | --- |
| **中文字段名** | **类型及长度** | **举例** |
| 固定配送路线编号 | char[6] | r00001 |
| 站点序号 | int | 1 |
| 站点编号 | char[10] | s00001 |
| 站点名称 | char[50] | 汉口北 |
| 与起始站点距离 | float | 10公里 |
| 与上一个站点距离 | float | 10公里 |
| 与上一个站点交通耗时 | float | 20分钟 |
| 停留耗时 | float | 2分钟 |
| 经过本站点固定路线编号，如为多条固定路线的交汇站点，需要给出多条固定路线编号 | char[50] | r00001 |

1. 车辆

|  |  |  |
| --- | --- | --- |
| **中文字段名** | **类型及长度** | **举例** |
| 车辆牌照 | char[8] | 鄂A00001 |
| 执行配送路线编号 | char[6] | r00001 |
| 司机姓名 | char[8] | 王宝强 |
| 司机移动电话 | char[11] | 14752344458 |

1. 货物

|  |  |  |
| --- | --- | --- |
| **中文字段名** | **类型及长度** | **举例** |
| 载货种类 | char[8] | 帽子 |
| 载货容量 | char[6] | 10.8 |
| 卸货种类 | char[8] | 裤子 |
| 卸货容量 | char[11] | 3.6 |

1. 三向十字链表结构

货物基本信息

货物基本信息

货物基本信息

head

路线1

路线2

路线n

NULL

**…**

站点1详细信息

站点2详细信息

站点n基本信息

经停站点1的车辆基本信息链

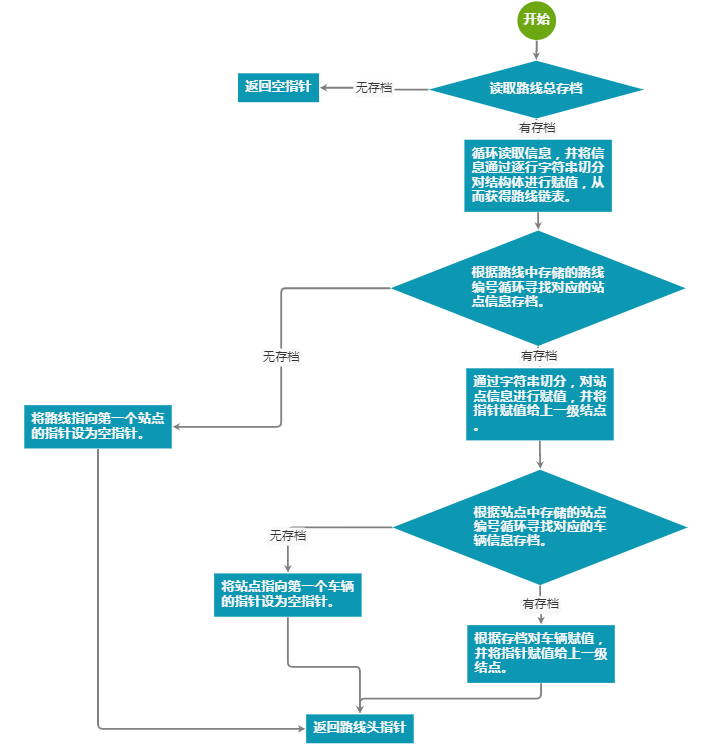
经停站点2的车辆基本信息链

经停站点n的车辆基本信息链

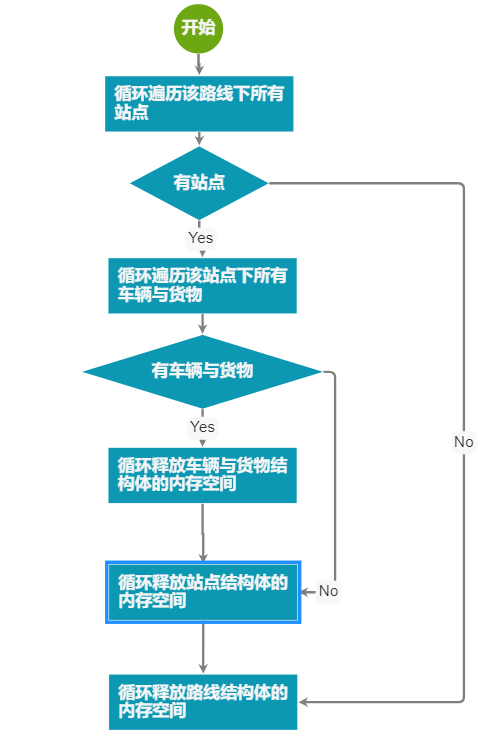
**…**

# 三．程序结构

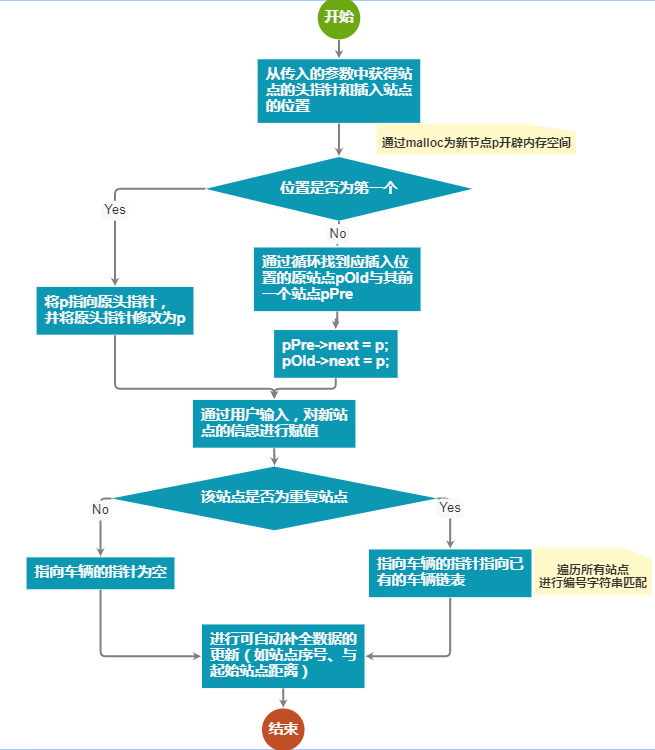
1.IO操作流程图（以加载所有存档为例）：



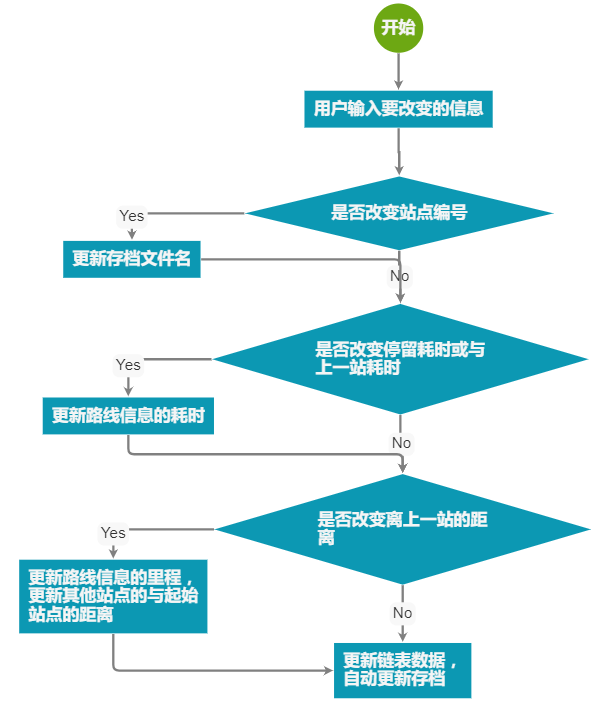
2.删除结点流程图（以删除路线结点为例）：



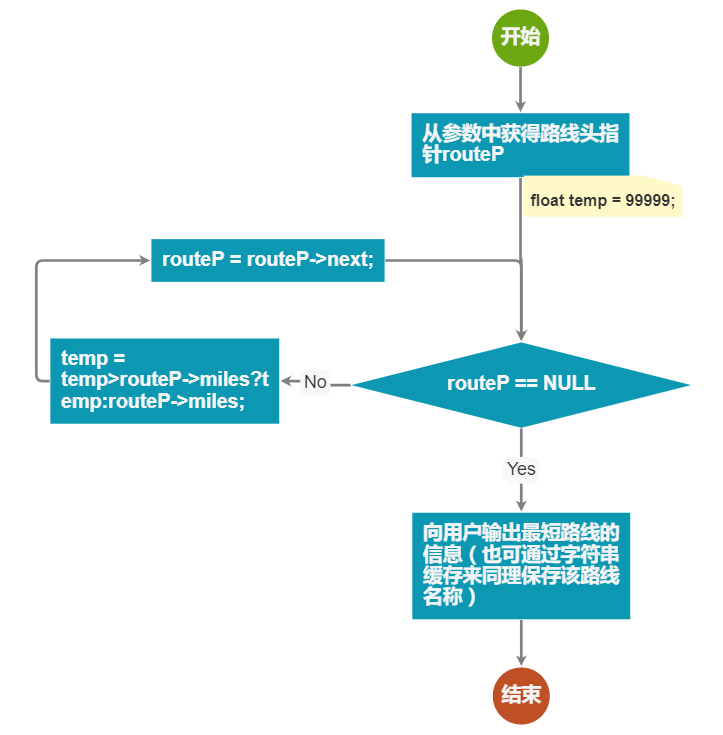
**3.插入路线结点流程：**



4.修改站点信息流程：



5.查询流程（以查询查询公里数最短的路线举例）：



6.生成统计报表流程：



# 四．各模块的功能

1.三种信息界面：

展示对应链表的信息报表，并在报表下方展示数字功能菜单。

2.快速查询界面：

展示所有可以查询的功能数字菜单，用户在选择后即可进行查询特定信息。

3.查看详细信息：

提供进入下一级链表信息的入口。

4.统计生成表格：

将全链表信息，以及统计数据信息写入csv文件，从而使用户可以通过Excel进行高级编辑。

5.自动更新存档：

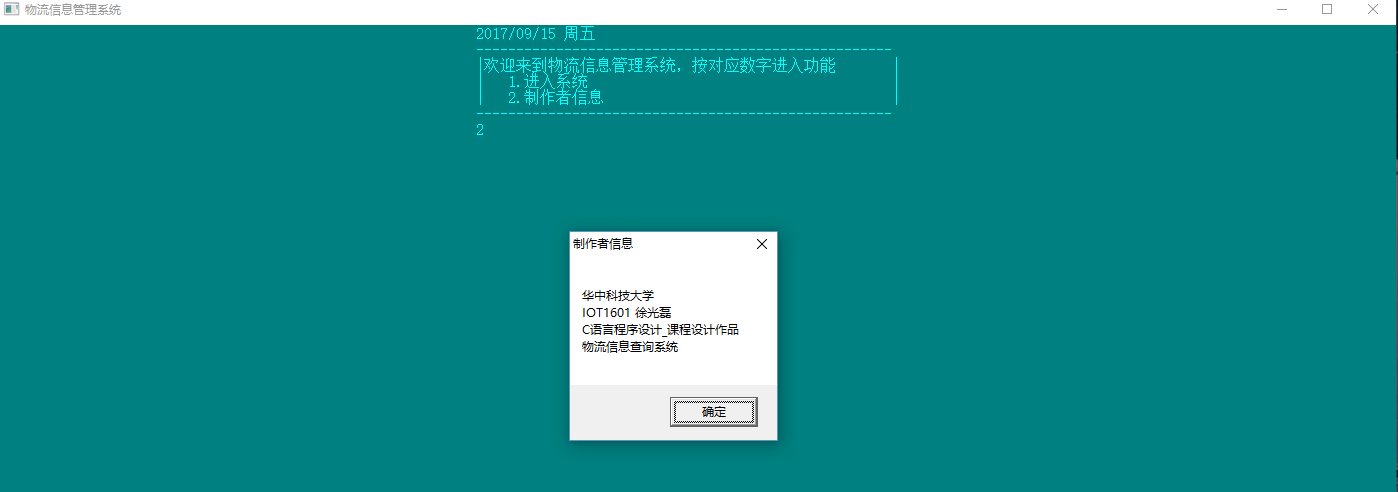
在变更链表内数据之后，对存档文件进行自动更新（根据不同需求选择不同的更新函数，避免不需要的更新文件以此提高速度）。

6.增删改操作：

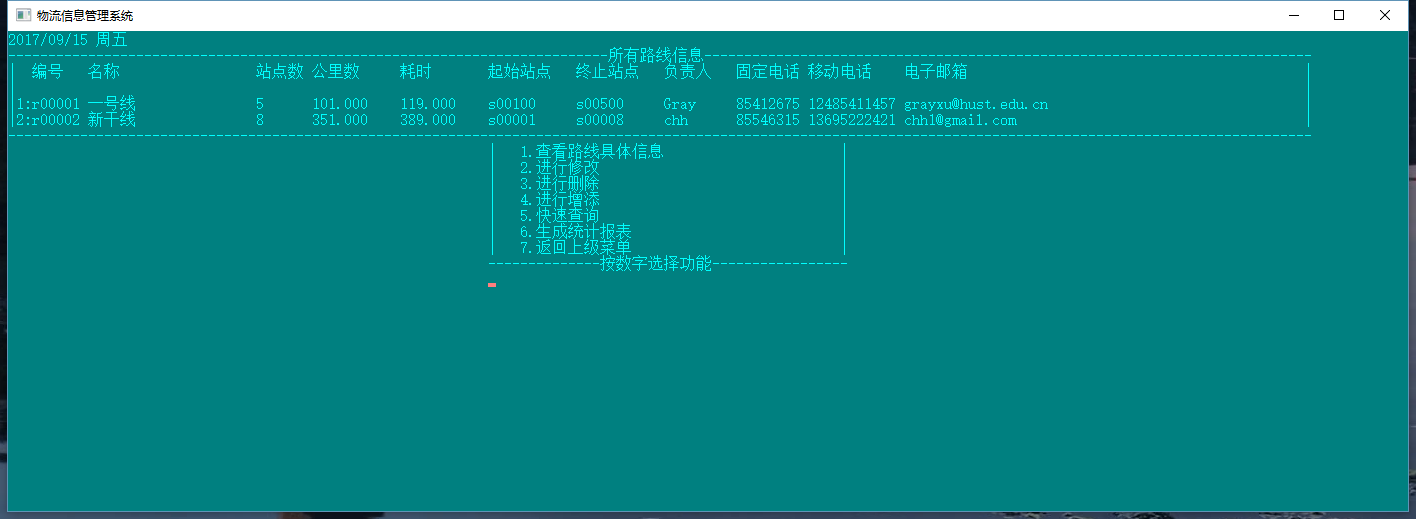
对三种不同链表的结点信息数据进行增删改操作，更新链表内数据。再根据改变的信息，自动搜索可以自动补全或更新的数据部分自动填入数据。

# 五．测试

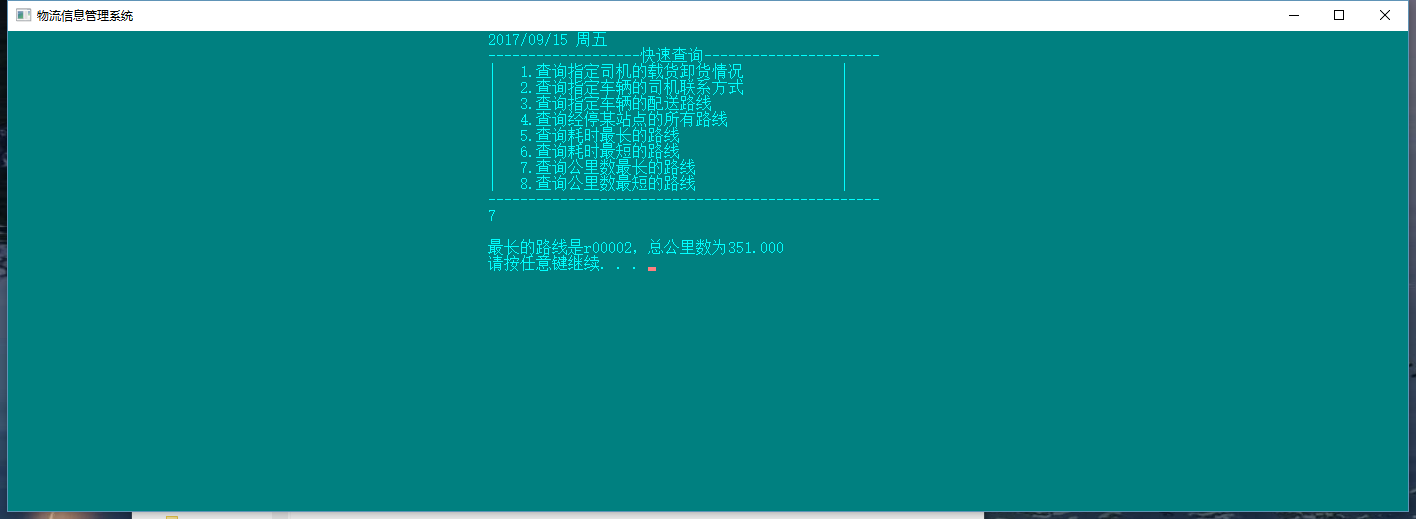
1.系统首页：



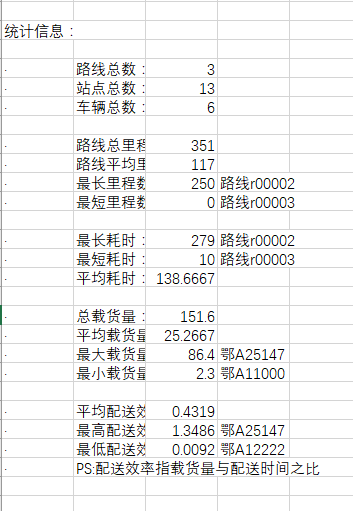
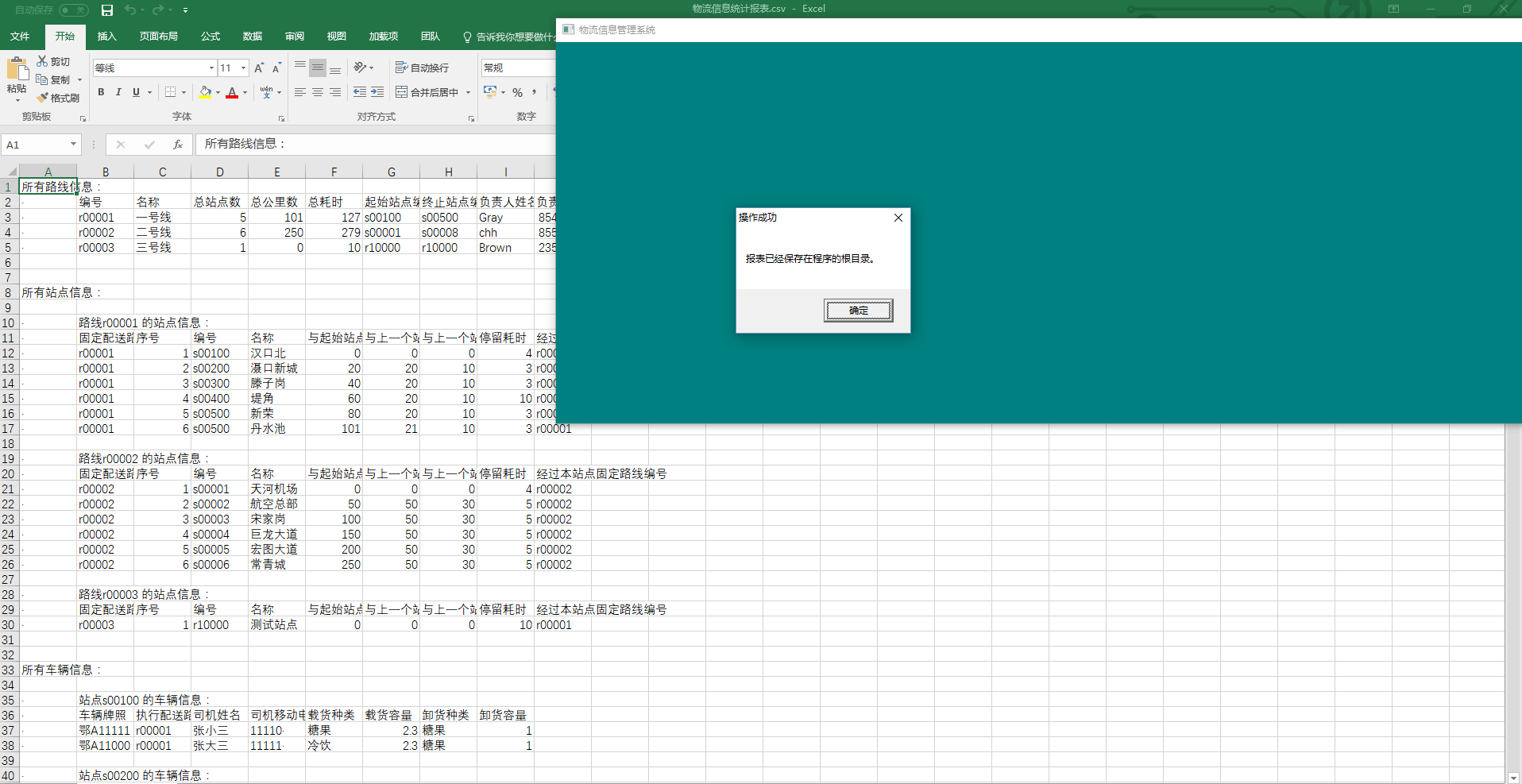
2.路线信息界面：



3.快速查询界面：



4.统计报表：



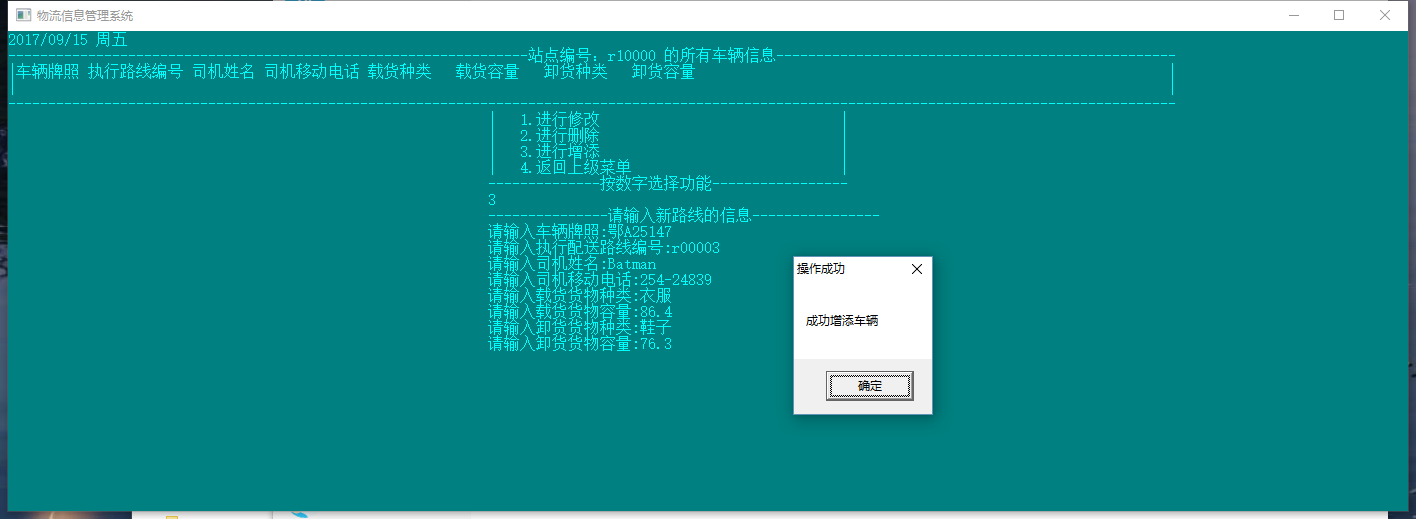
收集链表中所有信息，并进行统计，最后将数据输出到表格文件中。

5.修改信息界面（站点）：



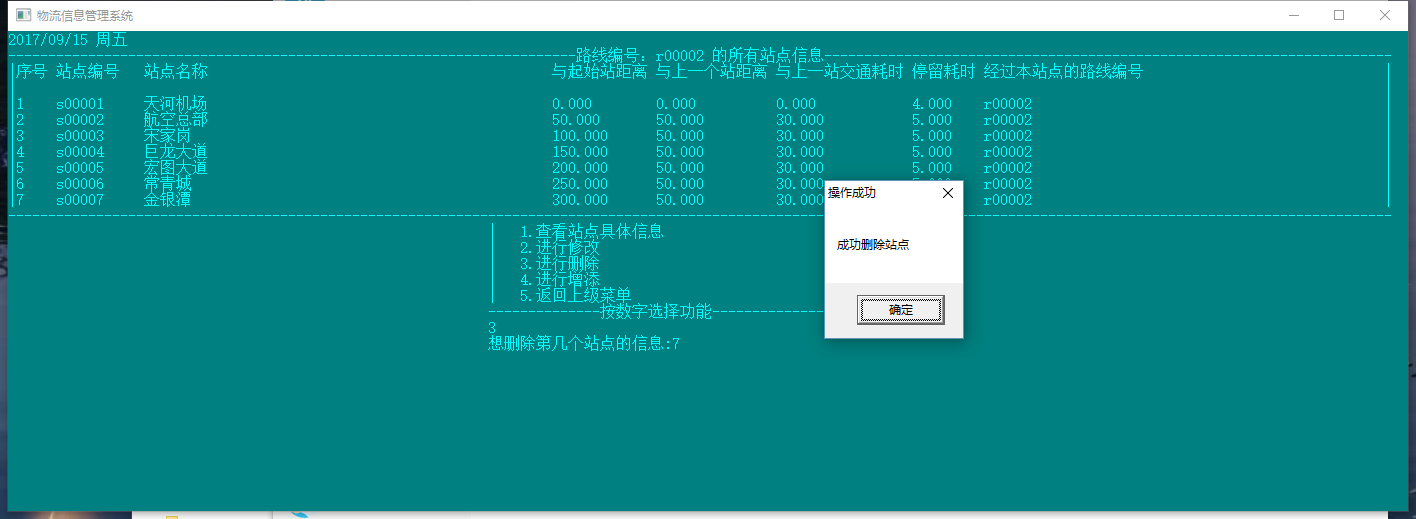
修改完毕后，其他站点信息、上一级路线信息联动地自动改变，并自动保存信息到存档内，防止数据遗失。

6.增添信息界面（车辆）：



增添数据能够判断哪些数据不需要手动输入而进行自动补全，增添后自动联动修改其他受到影响的数据（在有足够信息自动判断的前提下）。

7.删除信息界面（站点）：



删除结点信息，先释放该结点及子节点所占的所有内存空间，并删除存档文件，然后完成自动修改受影响的数据（在有足够信息自动判断的前提下）。最后更新数据到存档文件。

# 六．体会

在本次的课程设计中，让我感觉最遗憾的就是没有用c语言去实现一个更佳的GUI界面，而只是在控制台界面下进行粗略的优化控制台界面（即修改了标题颜色、对齐方式和数字菜单）。对于一个面向用户的软件，没有真正图形界面是一个大大的硬伤，影响用户体验，更影响这款应用的传播。我在规划的阶段，便开始了解c语言对GUI的支持。发现真正用C语言写GUI程序的很少很老，像Gtk对win32 api的包装思路就落伍于当前GUI开发的发展。类似Qt等包装较完善的库又需要进而掌握C++的各种特性，学习曲线之陡又让我望而生畏……不由得感叹C/C++在给开发者高自由度的同时，也降低了开发效率。

本次课程设计中，我并没有像给的模板一样去严格地分明几个模块的作用，这样的操作对于控制台程序来说，操作过于繁琐。复杂界面下的表现明显不够人性化。与之相反，我在打开程序的一开始便在界面中展示出所有的路线信息，然后直接提供再进一步的功能选项。这样的选择某种程度上也是在数字菜单下的一种妥协。

作为一个信息管理系统，明显的特征就是信息量大，处理信息多，所以我在系统中为可以自动补全的数据都进行了自动添加或更新的功能（如站点的序号，离终点站的距离、站点数等等），希望能以此提高用户的录入效率。这也是一种出于架构上的考虑。

本程序中我另一个面对的问题就是如何向用户直观地体现所有的信息，C语言并没有和Excel、SQL交互的开发库。于是我采用了将统计报表输入csv逗号分隔符文件中，既照顾到了C语言写入数据能力的薄弱，也照顾到了能使用Excel进行高级编辑的能力。

# 七．致谢

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. <https://docs.microsoft.com/en-us/windows/console/console-reference>

微软官网上的控制台操作文档。

1. 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。

# 附录：源代码清单

ListTool.h

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<DataStructure.h>

/\* CURD for routes \*/

/\* 创建储存路线数据的线性表\*/

route \*creatRouteList(FILE\* fText) {

if (feof(fText)) {

return NULL;

}

route \*routeP, \*pReturn;

//set the first node (of course this is not an elagent way to realize

routeP = (route \*)malloc(sizeof(route));

char StrLine[200];

fgets(StrLine, 200, fText);

char \*\* info = readRoute(StrLine);

if (info == NULL){//第一行就是空行

return NULL;

}

strcpy(routeP->routeID, info[0]);

strcpy(routeP->name, info[1]);

routeP->siteNums = atoi(info[2]);

routeP->miles = (float)atof(info[3]);

routeP->period = (float)atof(info[4]);

strcpy(routeP->startSite, info[5]);

strcpy(routeP->endSite, info[6]);

strcpy(routeP->adminName, info[7]);

strcpy(routeP->tel, info[8]);

strcpy(routeP->mTel, info[9]);

strcpy(routeP->email, info[10]);

freeDoubleCharArray(11, info);//deeply free

routeP->next = NULL;

pReturn = routeP;

//read other route info from text

while (1) {

route \* tempP = (route \*)malloc(sizeof(route));

char StrLine[200];

fgets(StrLine, 200, fText);

if (feof(fText)) {

free(tempP);

break;

}

char \*\* info = readRoute(StrLine);

strcpy(tempP->routeID, info[0]);

strcpy(tempP->name, info[1]);

tempP->siteNums = atoi(info[2]);

tempP->miles = (float) atof(info[3]);

tempP->period = (float)atof(info[4]);

strcpy(tempP->startSite, info[5]);

strcpy(tempP->endSite, info[6]);

strcpy(tempP->adminName, info[7]);

strcpy(tempP->tel, info[8]);

strcpy(tempP->mTel, info[9]);

strcpy(tempP->email, info[10]);

freeDoubleCharArray(11, info);//deeply free

routeP->next = tempP;

routeP = tempP;

routeP->next = NULL;

}

return pReturn;

}

//创建信息链表，返回指向该路线的第一个站点的指针，

//从而外部可以将站点连接上路线结点

site \* creatSiteList(FILE\* fText) {

site \*siteP, \*pReturn;

//set the first node (of course this is not an elagent way to realize this method

siteP = (site \*)malloc(sizeof(site));

char StrLine[200];

fgets(StrLine, 200, fText);

char \*\* info = readRoute(StrLine);

if (info == NULL){//第一行就是空行

return NULL;

}

strcpy(siteP->routeID, info[0]);

siteP->siteSID = atoi(info[1]);

strcpy(siteP->siteID, info[2]);

strcpy(siteP->siteName, info[3]);

siteP->d2Start = atof(info[4]);

siteP->d2Last = atof(info[5]);

siteP->time2Last = atof(info[6]);

siteP->waitTime = atof(info[7]);

strcpy(siteP->routeIDArray, info[8]);

freeDoubleCharArray(9, info);

siteP->next = NULL;

pReturn = siteP;

//read other site info from text

while (1) {

site \* tempP = (site \*)malloc(sizeof(site));

char StrLine[200];

fgets(StrLine, 200, fText);

if (feof(fText)) {

free(tempP);

break;

}

char \*\* info = readRoute(StrLine);//进行字符串切分操作

strcpy(tempP->routeID, info[0]);

tempP->siteSID = atoi(info[1]);

strcpy(tempP->siteID, info[2]);

strcpy(tempP->siteName, info[3]);

tempP->d2Start = atof(info[4]);

tempP->d2Last = atof(info[5]);

tempP->time2Last = atof(info[6]);

tempP->waitTime = atof(info[7]);

strcpy(tempP->routeIDArray, info[8]);

freeDoubleCharArray(9, info);

siteP->next = tempP;

siteP = tempP;

siteP->next = NULL;

}

return pReturn;

}

car \* creatCarList(FILE \* fText) {

car \*carP, \*pReturn;

//set the first node (of course this is not an elagent way to realize this method

carP = (car \*)malloc(sizeof(car));

char StrLine[200];

StrLine[0] = '\0';

fgets(StrLine, 200, fText);

char \*\* info = readCar(StrLine);

if (info == NULL){//第一行就是空行

return NULL;

}

strcpy(carP->carID, info[0]);

strcpy(carP->routeID, info[1]);

strcpy(carP->driverName, info[2]);

strcpy(carP->driverTel, info[3]);

freeDoubleCharArray(4, info);

carP->next = NULL;

pReturn = carP;

//read other site info from text

while (1) {

car \* tempP = (car \*)malloc(sizeof(car));

char StrLine[200];

fgets(StrLine, 200, fText);

if (feof(fText)) {

free(tempP);

break;

}

char \*\* info = readCar(StrLine);//进行字符串切分操作

strcpy(tempP->carID, info[0]);

strcpy(tempP->routeID, info[1]);

strcpy(tempP->driverName, info[2]);

strcpy(tempP->driverTel, info[3]);

freeDoubleCharArray(4, info);

carP->next = tempP;

carP = tempP;

carP->next = NULL;

}

return pReturn;

}

good\* creatGood(FILE \* fText) {

good\* goodP = (good \*)malloc(sizeof(good));

char StrLine[200];

fgets(StrLine, 200, fText);

char \*\* info = readGood(StrLine);

if (info == NULL){

return NULL;

}

strcpy(goodP->uploadType, info[0]);

goodP->upVolume = atof(info[1]);

strcpy(goodP->downloadType, info[2]);

goodP->downVolume = atof(info[3]);

freeDoubleCharArray(4, info);

return goodP;

}

/\* 打印链表，链表的遍历，TODO：需要做格式化，需要深度遍历(报表？？)\*/

void printList(route \*pHead) {

// TODO

}

/\* 返回链表的长度\*/

int sizeRouteList(route \*pHead) {

int size = 0;

while (pHead != NULL) {

size++;

pHead = pHead->next;

}

return size;

}

int sizeSiteList(site \*pHead) {

int size = 0;

while (pHead != NULL) {

size++;

pHead = pHead->next;

}

return size;

}

int sizeCarList(car \*pHead) {

int size = 0;

while (pHead != NULL) {

size++;

pHead = pHead->next;

}

return size;

}

/\* 获得某位置结点的地址，让外部进行修改, 失败则返回空指针, 从0记起\*/

route\* getRoutePointer(route \*pHead, int pos) {

if (pHead == NULL) {

return NULL;

}

route \*pNode = pHead;

int i = 0;

while (pNode != NULL) {

if (i == pos) {

break;

}

pNode = pNode->next;

i++;

}

return pNode;

}

site \*getSitePointer(site \*pHead, int pos) {

if (pHead == NULL) {

return NULL;

}

site \*pNode = pHead;

int i = 0;

while (pNode != NULL) {

if (i == pos) {

break;

}

pNode = pNode->next;

i++;

}

return pNode;

}

car \*getCarPointer(car \*pHead, int pos) {

if (pHead == NULL) {

return NULL;

}

car \*pNode = pHead;

int i = 0;

while (pNode != NULL) {

if (i == pos) {

break;

}

pNode = pNode->next;

i++;

}

return pNode;

}

/\* 向单链表中第pos个结点位置插入元素为x的结点(把该位置的往后挤)，若插入成功返回新结点的指针，否则返回NULL\*/

route\* AddRouteNode(route \*HEAD, int pos) {

route \*pHead = HEAD;

route \*pPre = HEAD;//both set to the first( init)

route \*pNew = NULL;

int i = 0;

pNew = (route \*)malloc(sizeof(route));

if (pPre == NULL) {

return NULL;

}

if (NULL == pHead) {

return NULL;

}

if (pos == 0) {//add to be the first one

pNew->next = HEAD;

} else if (pos == sizeRouteList(HEAD)) {//add to be the last one

route \* lastRouteP = getRoutePointer(HEAD, pos - 1);

lastRouteP->next = pNew;

pNew->next = NULL;

} else {

while (pHead != NULL) {

if (i == pos)

break;

pPre = pHead;

pHead = pHead->next;

++i;

}

pPre->next = pNew;

pNew->next = pHead;

}

return pNew;

}

site \* AddSiteNode(site \*HEAD, int pos) {

site \*pHead = HEAD;

site \*pPre = HEAD;//both set to the first( init)

site \*pNew = NULL;

int i = 0;

pNew = (site \*)malloc(sizeof(site));

if (pPre == NULL) {

return NULL;

}

if (pos == 0) {//add to be the first one

pNew->next = HEAD;

} else if (pos == sizeSiteList(HEAD)) {//add to be the last one

site \* lastSiteP = getSitePointer(HEAD, pos - 1);

lastSiteP->next = pNew;

pNew->next = NULL;

} else {

while (pHead != NULL) {

if (i == pos)

break;

pPre = pHead;

pHead = pHead->next;

++i;

}

pPre->next = pNew;

pNew->next = pHead;

}

return pNew;

}

car \* AddCarNode(car \*HEAD, int pos) {

car \*pHead = HEAD;

car \*pPre = HEAD;//both set to the first( init)

car \*pNew = NULL;

int i = 0;

pNew = (car \*)malloc(sizeof(car));

if (pPre == NULL) {

return NULL;

}

if (NULL == pHead) {

return NULL;

}

if (pos == 0) {//add to be the first one

pNew->next = HEAD;

} else if (pos == sizeCarList(HEAD)) {//add to be the last one

car \* lastSiteP = getCarPointer(HEAD, pos - 1);

lastSiteP->next = pNew;

pNew->next = NULL;

} else {

while (pHead != NULL) {

if (i == pos)

break;

pPre = pHead;

pHead = pHead->next;

++i;

}

pPre->next = pNew;

pNew->next = pHead;

}

return pNew;

}

//With using this method you can free all info of this routeP in the big list.

void deeplyFreeRoute(route \* routeP){

site\* siteP = routeP->firstSite;

site\* sitePp = NULL;

while (siteP != NULL){

sitePp = siteP->next;

car\* carP = siteP->carHeadP;

car\* carPp = NULL;

while (carP != NULL){

carPp = carP->next;

free(carP->good);

free(carP);

carP = carPp;

}

free(siteP);

siteP = sitePp;

}

}

/\* 从单链表中删除第pos个结点(兼容头尾), @return: the new head pointer.\*/

route\* DelRoutePos(route \*HeadP, int pos) {

route \*pHead = HeadP;

route \*pTmp = HeadP;//pTmep即为被删除结点的前一个结点

//防空指针

if (NULL == pHead) {

printf("DelPos函数执行，链表为空\n");

return NULL;

}

register int i = 0;

while (pHead != NULL) {

if (i == pos)

break;

pTmp = pHead;

pHead = pHead->next;

++i;

}

if (i==0) {//即删除头结点

route \* returnP = pHead->next;//返回第二个结点

deeplyFreeRoute(pHead);

return returnP;

}

pTmp->next = pHead->next;

deeplyFreeRoute(pHead);

return HeadP;

}

site\* DelSitePos(site \*HeadP, int pos) {

site \*pHead = HeadP;

site \*pTmp = HeadP;//pTmep即为被删除结点的前一个结点

//防空指针

car\* carP = NULL;//内部结点free，防止内存泄露

car\* carPp = NULL;

if (NULL == pHead) {

printf("DelPos函数执行，链表为空\n");

return NULL;

}

register int i = 0;

while (pHead != NULL) {

if (i == pos)

break;

pTmp = pHead;

pHead = pHead->next;

++i;

}

if (pHead == NULL){//pos超界，寻找不到对应的结点

printf("输入指令错误\n");

return HeadP;

}

if (i == 0) {//即删除头结点

site \* returnP = pHead->next;//返回第二个结点

//deeply free

carP = pHead->carHeadP;

while(carP != NULL){

carPp = carP->next;

free(carP->good);

free(carP);

carP = carPp;

}

free(pHead);

return returnP;

}

pTmp->next = pHead->next;

//deeply free

carP = pHead->carHeadP;

while(carP != NULL){

carPp = carP->next;

free(carP->good);

free(carP);

carP = carPp;

}

free(pHead);

return HeadP;

}

car\* DelCarPos(car \*HeadP, int pos) {

car \*pHead = HeadP;

car \*pTmp = HeadP;//pTmep即为被删除结点的前一个结点

//防空指针

if (NULL == pHead) {

printf("DelPos函数执行，链表为空\n");

return NULL;

}

register int i = 0;

while (pHead != NULL) {

if (i == pos)

break;

pTmp = pHead;

pHead = pHead->next;

++i;

}

if (i == 0) {//即删除头结点

car \* returnP = pHead->next;//返回第二个结点

free(pHead->good);

free(pHead);

return returnP;

}

pTmp->next = pHead->next;

free(pHead->good);

free(pHead);

return HeadP;

}

/\* 交换2个元素的位置，记得检测头是否改变 \*/

void swapRoute(route \*\*ppNode, int posA, int posB) {

route \*node = \*ppNode;

int i;

route \*preAp = NULL;

route \*preBp = NULL;

route \*Ap = NULL;

route \*Bp = NULL;

for (i = 0; i < sizeRouteList(node); i++) {

if (i == posA) {

Ap = node;

}

else if (i == posB) {

Bp = node;

}

else if (i == posA - 1) {

preAp = node;

}

else if (i == posB - 1) {

preBp = node;

}

node = node->next;

}

if (preBp != NULL && preAp != NULL) {

//they are both not the first one

route \*tempP = preAp->next;

preAp->next = preAp->next;

preBp->next = tempP->next;

tempP = Ap->next;

Ap->next = Bp->next;

Bp->next = tempP;

}

else if (preAp == NULL) {//A is the first one

route \*tempP = Ap->next;

Ap->next = Bp->next;

Bp->next = tempP;

preBp->next = Ap;

}

else {//preBp == NULL

route \*tempP = Ap->next;

Ap->next = Bp->next;

Bp->next = tempP;

preAp->next = Bp;

}

}

DataStructure.h

#include <stdlib.h>

char\*\* setDoubleCharArray(char\*\* out, char\* in);

/\*\*

\* 货物的基本信息：\*Type : 装货、卸货的货物的种类

\* \*Volume : 货物所占容量，（单位立方米）

\*/

typedef struct Good {

char uploadType[10];

float upVolume;

char downloadType[10];

float downVolume;

} good;

char\*\* readGood(char\* in) {

char\*\* strArray = (char\*\*)malloc(sizeof(char\*) \* 4);

strArray[0] = (char\*)malloc(sizeof(char) \* 10);

strArray[1] = (char\*)malloc(sizeof(char) \* 10);

strArray[2] = (char\*)malloc(sizeof(char) \* 10);

strArray[3] = (char\*)malloc(sizeof(char) \* 10);

if (setDoubleCharArray(strArray, in) != NULL){

return strArray;

} else {

return NULL;

}

}

/\*\*

\* 配送车辆的基本信息： carID： 车辆牌照

\* routeID：执行配送路线编号

\* driverName： 司机姓名

\* driverTel： 司机移动电话

\* carID: 车辆编号

\* goodList： 运货清单

\*/

typedef struct Car {

char carID[9];

char routeID[7];

char driverName[9];

char driverTel[12];

char siteID[11];

good \* good;

struct Car \* next;

} car;

char \*\* readCar(char\* in) {

char\*\* strArray = (char\*\*)malloc(sizeof(char\*) \* 4);

strArray[0] = (char\*)malloc(sizeof(char) \* 9);

strArray[1] = (char\*)malloc(sizeof(char) \* 7);

strArray[2] = (char\*)malloc(sizeof(char) \* 9);

strArray[3] = (char\*)malloc(sizeof(char) \* 12);

if (setDoubleCharArray(strArray, in) != NULL){

return strArray;

} else {

return NULL;

}

}

/\*\*

\* 经停站点详细信息: routeNums: 固定配送路线数量

\* routeIDArray: 固定配送路线编号的数组

\* siteSID: 站点序号

\* siteID: 站点编号

\* siteNames: 站点名称

\* d2Start: 与初始站点距离

\* d2Last: 与上一个站点距离

\* t2Last: 与上一个站点交通耗时

\* tWait: 停留耗时

\* next: 指向下一个站点的指针

\*/

typedef struct Site {

char routeID[7];

int siteSID;

char siteID[11];

char siteName[51];

float d2Start;

float d2Last;

float time2Last;

float waitTime;

char routeIDArray[50];

struct Car \* carHeadP;

struct Site \* next;

} site;

char\*\* readSite(char\* in) {

char\*\* strArray = (char\*\*)malloc(sizeof(char \*) \* 9);

strArray[0] = (char\*)malloc(sizeof(char) \* 7);

strArray[1] = (char\*)malloc(sizeof(char) \* 11);

strArray[2] = (char\*)malloc(sizeof(char) \* 11);

strArray[3] = (char\*)malloc(sizeof(char) \* 51);

strArray[4] = (char\*)malloc(sizeof(char) \* 11);

strArray[5] = (char\*)malloc(sizeof(char) \* 11);

strArray[6] = (char\*)malloc(sizeof(char) \* 11);

strArray[7] = (char\*)malloc(sizeof(char) \* 11);

strArray[8] = (char\*)malloc(sizeof(char) \* 51);

if (setDoubleCharArray(strArray, in) != NULL){

return strArray;

} else {

return NULL;

}

}

/\*\*

\* 配送路线详细信息: routeID: 固定配送路线编号

\* name: 固定配送路线编号名称

\* siteNums: 固定配送路线总站点数

\* miles: 固定配送路线总公里数

\* period: 全站点配送总耗时

\* startSite: 起始站点编号

\* endSite: 终止站点编号

\* adminName: 负责人姓名

\* tel: 负责人办公室电话

\* mTel: 负责人移动电话

\* email: 负责人电子邮箱

\* firstSite: 指向站点（即第一个站点）的指针

\* next: 指向下一个路线的指针

\*/

typedef struct Route {

char routeID[7];

char name[21];

short siteNums;

float miles;

float period;

char startSite[11];

char endSite[11];

char adminName[9];

char tel[9];

char mTel[12];

char email[51];

struct Site \* firstSite;//指向该路线的第一个站点，并形成路线链表

struct Route \* next;

} route;

/\*把数据分解为二维字符指针\*/

char\*\* readRoute(char\* in) {

char\*\* strArray = (char\*\*)malloc(sizeof(char \*) \* 11);

strArray[0] = (char\*)malloc(sizeof(char) \* 7);

strArray[1] = (char\*)malloc(sizeof(char) \* 21);

strArray[2] = (char\*)malloc(sizeof(char) \* 11);

strArray[3] = (char\*)malloc(sizeof(char) \* 11);

strArray[4] = (char\*)malloc(sizeof(char) \* 11);

strArray[5] = (char\*)malloc(sizeof(char) \* 11);

strArray[6] = (char\*)malloc(sizeof(char) \* 11);

strArray[7] = (char\*)malloc(sizeof(char) \* 9);

strArray[8] = (char\*)malloc(sizeof(char) \* 9);

strArray[9] = (char\*)malloc(sizeof(char) \* 12);

strArray[10] = (char\*)malloc(sizeof(char) \* 51);

if (setDoubleCharArray(strArray, in) != NULL){

return strArray;

} else {

return NULL;

}

}

//free stake memory( for the double char array)

void freeDoubleCharArray(int num, char \*\* info) {

register int i;

for (i = 0; i < num; i++) {

free(info[i]);

}

free(info);

}

//将一维字符数组的内容切分至已经开好空间的二维字符数组中去。(仅适用于必须有内容的情况

char\*\* setDoubleCharArray(char\*\* out, char\* in) {

if (out == NULL || in[0]=='\0') {

return NULL;

}

register int allIndex = 0;

register int outIndex = 0;

register int inIndex = 0;

char read = 0;

while ((read = \*(in + allIndex)), read != '\n') {

if (read == ';') {

out[outIndex][inIndex] = '\0';//end this string

outIndex++;//reset

inIndex = 0;

} else {

out[outIndex][inIndex] = read;

inIndex++;

}

allIndex++;

}

out[outIndex][inIndex] = '\0';//end the final string

return out;

}

Main.c

#include <stdio.h>

#include <ListTool.h>

#include <stdlib.h>

#include <conio.h>

#include <windows.h>

#include <wincon.h>

#include <windef.h>

#include "shlobj.h"

#include <wingdi.h>

//TODO: 报告

route \* initData();

void printSomeSpace(int num);

void printSomeG(int num);

void printFronPage();

void printPowerBy();

void printRoutePage(route \* routeHeadP);

void printSitePage(route \*routeP, route \* routeHeadP);

void printCarPage(site \* siteP);

void quickQuery(route \* routeHeadP);

void addRoute(route \* routeHeadP);

car\* addCar(car \* carHeadP);

site\* addSite(site \* siteHeadP, char\* routeIDIN, route \* routeHeadP);

int changeSite(site \* siteSpecific, int notFirst, site \* siteHeadP);

int changeRoute(route \* routeSpecial);

int changeCar(car \* carSpecific);

void updateCarsFILE(car \* carHeadP, char \* fileName);

void updateRoutesFILE(route\* routeHeadP);

void updateSitesFILE(site \* siteHeadP);

char\* noNfgets(char \* Buffer, int MaxConut, FILE\* Stream);

void changeOldRouteID(route \* routeSpecific);

int changeSaveName(char\* old, char\* new);

void updateSitesCount(route \* routeSpecific);

void initConsole();

void makeCsv(route \* routeHeadP);

float totalTime(route \*routeP);

float getMile(char \*routeIDThis, route \* routeHeadP);

float totalMile(route \*routeP);

void updateAlld2S(site \* siteP);

int main() {

route\* routeHeadP;

routeHeadP = initData();//初始化数据入链表

char \*url = malloc(sizeof(char) \* 20);//开辟空间待使用

int space = 60;

initConsole();

//主函数里包含了第一层信息展示

while (1) {

printFronPage();

int input1 = 0;

scanf("%d%\*c", &input1);

if (input1 == 1) {

int isRoutePage = 1;

while (isRoutePage) {

printRoutePage(routeHeadP);//输出路线页的信息

int input2 = 0; scanf("%d%\*c", &input2);

int seq = 0;

if (input2 == 1) {

printSomeSpace(space);printf("想详细查看第几条路线的站点信息：");

seq = 0; scanf("%d%\*c", &seq);

route \* routeSpecific = getRoutePointer(routeHeadP, seq - 1);

if (routeSpecific != NULL){

printSitePage(getRoutePointer(routeHeadP, seq - 1), routeHeadP);//进入该函数内部

} else {

printSomeSpace(space);printf("错误输入\n");

}

} else if (input2 == 2) {

printSomeSpace(space);printf("想修改第几条路线：");

seq = 0; scanf("%d%\*c", &seq);

changeRoute(getRoutePointer(routeHeadP, seq-1));

updateRoutesFILE(routeHeadP);

MessageBox(NULL, TEXT("成功修改路线"), TEXT("操作成功"), MB\_OK);

system("cls");

} else if (input2 == 3) {

printSomeSpace(space);printf("想删除第几条路线：");

seq = 0; scanf("%d%\*c", &seq);

//删除对应的存档文件 TODO：深层筛选未重复站点删除

strcpy(url, "save/");

strcat(url, getRoutePointer(routeHeadP, seq - 1)->routeID);

strcat(url, ".txt");

remove(url);

//再更新路线合集存档

routeHeadP = DelRoutePos(routeHeadP, seq-1);

updateRoutesFILE(routeHeadP);

MessageBox(NULL, TEXT("成功删除路线"), TEXT("操作成功"), MB\_OK);

system("cls");

} else if (input2 == 4) {

addRoute(routeHeadP);

updateRoutesFILE(routeHeadP);

MessageBox(NULL, TEXT("成功增添路线\n建议继续进入该路线详细信息界面添加站点信息"), TEXT("操作成功"), MB\_OK);

system("cls");

} else if (input2 == 5) {

//快速查询命令

quickQuery(routeHeadP);

system("cls");

} else if (input2 == 6){

makeCsv(routeHeadP);//生成统计报表

system("cls");

} else if (input2 == 7){

isRoutePage = 0;

system("cls");

}

}

} else if (input1 == 2) {

printPowerBy();

continue;

} else {

system("cls");

printSomeSpace(space);printf("输入错误指令\n");

}

}

free(url);

return 0;

}

//快速查询实现逻辑

void quickQuery(route \* routeHeadP){

route \* routeP = routeHeadP;

system("cls");

printSomeSpace(60);system("date /T");

printSomeSpace(60);printf("-------------------快速查询----------------------\n");

printSomeSpace(60);printf("|\t1.查询指定司机的载货卸货情况\t\t|\n");

printSomeSpace(60);printf("|\t2.查询指定车辆的司机联系方式\t\t|\n");

printSomeSpace(60);printf("|\t3.查询指定车辆的配送路线\t\t|\n");

printSomeSpace(60);printf("|\t4.查询经停某站点的所有路线\t\t|\n");

printSomeSpace(60);printf("|\t5.查询耗时最长的路线\t\t\t|\n");

printSomeSpace(60);printf("|\t6.查询耗时最短的路线\t\t\t|\n");

printSomeSpace(60);printf("|\t7.查询公里数最长的路线\t\t\t|\n");

printSomeSpace(60);printf("|\t8.查询公里数最短的路线\t\t\t|\n");

printSomeSpace(60);printf("-------------------------------------------------\n");

int seq = 0;printSomeSpace(60);scanf("%d%\*c",&seq);

char input[21];

float tempFloat = 0;

int isSuccessful = 0;

printf("\n");

printSomeSpace(60);

switch (seq){

case 1:

printf("请输入司机姓名：");

noNfgets(input, 20, stdin);

l4:while (routeP != NULL){

site\* siteP = routeP->firstSite;

while (siteP != NULL){

car\* carP = siteP->carHeadP;

while (carP != NULL){

if(strcmp(input, carP->driverName) == 0){

printSomeSpace(60);

if (carP->good != NULL){

printf("司机 %s载货了容量为%.3f的%s，卸货了容量为%.3f的%s\n", input, carP->good->upVolume, carP->good->uploadType, carP->good->downVolume, carP->good->downloadType);

} else {

printf("该司机无货物信息\n");

}

isSuccessful = 1;

carP = NULL;

siteP = NULL;

routeP = NULL;

goto l4;//jump out from loops

}

carP = carP->next;

}

siteP = siteP->next;

}

routeP = routeP->next;

}

break;

case 2:

printf("请输入车辆牌照：");

noNfgets(input, 20, stdin);

l3:while (routeP != NULL){

site\* siteP = routeP->firstSite;

while (siteP != NULL){

car\* carP = siteP->carHeadP;

while (carP != NULL){

if (strcmp(carP->carID, input) == 0){

printSomeSpace(60);printf("车辆%s的司机%s的联系方式为%s\n", input, carP->driverName, carP->driverTel);

isSuccessful = 1;

carP = NULL;

siteP = NULL;

routeP = NULL;

goto l3;//jump out from loops

}

carP = carP->next;

}

siteP = siteP->next;

}

routeP = routeP->next;

}

break;

case 3:

printf("请输入车辆牌照：");

noNfgets(input, 20, stdin);

l2:while (routeP != NULL){

site\* siteP = routeP->firstSite;

while (siteP != NULL){

car\* carP = siteP->carHeadP;

while (carP != NULL){

if (strcmp(carP->carID, input) == 0){

printSomeSpace(60);printf("车辆%s的配送路线为%s\n", input, carP->routeID);

isSuccessful = 1;

carP = NULL;

siteP = NULL;

routeP = NULL;

goto l2;//jump out from loops

}

carP = carP->next;

}

siteP = siteP->next;

}

routeP = routeP->next;

}

break;

case 4:

printf("请输入站点编号：");

noNfgets(input, 20, stdin);

l1:while (routeP != NULL){

site \* siteP = routeP->firstSite;

while (siteP != NULL){

if (strcmp(siteP->siteID, input) == 0){

printSomeSpace(60);printf("经停站点%s的所有路线:%s\n", input, siteP->routeIDArray);

isSuccessful = 1;

siteP = NULL;

routeP = NULL;//jump out from loops

goto l1;

}

siteP = siteP->next;

}

routeP = routeP->next;

}

break;

case 5:

tempFloat = -1;

while (routeP != NULL){

if (routeP->period > tempFloat){

tempFloat = routeP->period;

strcpy(input, routeP->routeID);

isSuccessful = 1;

}

routeP = routeP->next;

}

printf("耗时最长的路线是%s，总耗时为%.3f\n", input, tempFloat);

break;

case 6:

tempFloat = 99999;

while (routeP != NULL){

if (routeP->period < tempFloat){

tempFloat = routeP->period;

strcpy(input, routeP->routeID);

isSuccessful = 1;

}

routeP = routeP->next;

}

printf("耗时最短的路线是%s，总耗时为%.3f\n", input, tempFloat);

break;

case 7:

tempFloat = -1;

while (routeP != NULL){

if (routeP->miles > tempFloat){

tempFloat = routeP->miles;

strcpy(input, routeP->routeID);

isSuccessful = 1;

}

routeP = routeP->next;

}

printf("最长的路线是%s，总公里数为%.3f\n", input, tempFloat);

// system("pause");

break;

case 8:

tempFloat = 99999;

while (routeP != NULL){

if (routeP->miles < tempFloat){

tempFloat = routeP->miles;

strcpy(input, routeP->routeID);

isSuccessful = 1;

}

routeP = routeP->next;

}

printf("最短的路线是%s，总公里数为%.3f\n", input, tempFloat);

break;

default:

break;

}

if (!isSuccessful){

printSomeSpace(60);printf("找不到相关信息\n");

}

printSomeSpace(60);system("pause");

}

//初始化控制台，进行一些个性化设置

void initConsole(){

// HWND hwnd=GetForegroundWindow();

system("mode con:cols=175 lines=30");

system("color 3B");

SetConsoleTitle("物流信息管理系统");

}

/\*初始化数据，读取存档\*/

route \* initData() {

FILE \*fRouteP = fopen("save/routes.txt", "r");

route \* routeHeadP = NULL;

if (fRouteP == NULL) {

printf ("initData error.");

return NULL;

} else {

//test

routeHeadP = creatRouteList(fRouteP);

fclose(fRouteP);

char \*url = malloc(sizeof(char) \* 20);

register int i = 0;

register int j = 0;

register int k = 0;

for (i = 0; i < sizeRouteList(routeHeadP); i++) {

route \*routeP = getRoutePointer(routeHeadP,i);

strcpy(url, "save/");

strcat(url, routeP->routeID);

strcat(url, ".txt");

FILE \*fSite = fopen(url, "r+");

if (fSite == NULL) {

routeP->firstSite = NULL;

continue;

}

site \* siteHeadP = creatSiteList(fSite);

fclose(fSite);

routeP->firstSite = siteHeadP;//将站点链表的头结点指针传递给对应的路线

//set infomation about those cars in this site

for (j = 0; j < sizeSiteList(siteHeadP); j++) {

site \* siteP = getSitePointer(siteHeadP, j);

strcpy(url, "save/");

strcat(url, siteP->siteID);

strcat(url, ".txt");

FILE \* fCar = fopen(url, "r+");

if (fCar == NULL) {

siteP->carHeadP = NULL;

continue;//连车的存档都没有，更别说货物了

}

car \* carHeadP = creatCarList(fCar);

siteP->carHeadP = carHeadP;

for (k = 0; k < sizeCarList(carHeadP); k++) {

car \* carP = getCarPointer(carHeadP, k);

strcpy(url, "save/");

strcat(url, carP->carID);

strcat(url, ".txt");

FILE \* fGood = fopen(url, "r+");

if (fGood == NULL) {

carP->good = NULL;

continue;

}

carP->good = creatGood(fGood);

fclose(fGood);

}

fclose(fCar);

}

fclose(fSite);

}

}

return routeHeadP;

}

//输出首页

void printFronPage() {

printSomeSpace(60);

system("date /T");printSomeSpace(60);

printf("----------------------------------------------------\n");printSomeSpace(60);

printf("|欢迎来到物流信息管理系统，按对应数字进入功能\t|\n");printSomeSpace(60);

printf("|\t1.进入系统\t\t\t\t\t|\n");printSomeSpace(60);

printf("|\t2.制作者信息\t\t\t\t\t|\n");printSomeSpace(60);

printf("----------------------------------------------------\n");printSomeSpace(60);

}

//查看制作者信息逻辑

void printPowerBy() {

MessageBox(NULL, TEXT("华中科技大学\nIOT1601 徐光磊\nC语言程序设计\_课程设计作品\n物流信息查询系统"), TEXT("制作者信息"), MB\_OK);

system("cls");

}

//输出路线信息

void printRoutePage(route \* routeHeadP) {

route \* routeP = routeHeadP;

system("cls");

system("date /T");

printSomeG(75);printf("所有路线信息");printSomeG(76);printf("\n");//title

char first[] = "编号";

char second[] = "名称";

char third[] = "站点数";

char fourth[] = "公里数";

char fifth[] = "耗时";

char sixth[] = "起始站点";

char seventh[] = "终止站点";

char eighth[] = "负责人";

char ninth[] = "固定电话";

char tenth[] = "移动电话";

char eleventh[] = "电子邮箱";

printf("| %-6s %-20s %-6s %-10s %-10s %-10s %-10s %-8s %-8s %-11s %-50s|\n", first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth, eleventh);

printf("|");printSomeSpace(161);printf("|\n");

register int i = 1;

while (routeP != NULL) {

printf("|%d:",i);

printf("%-6s %-20s %-6d %-10.3f %-10.3f %-10s %-10s %-8s %-8s %-11s %-50s|\n", routeP->routeID, routeP->name, routeP->siteNums, routeP->miles, routeP->period, routeP->startSite, routeP->endSite, routeP->adminName, routeP->tel, routeP->mTel, routeP->email);

routeP = routeP->next;

i++;

}

int space = 60;

printSomeG(163);

printf("\n");

printSomeSpace(space);

printf("|\t1.查看路线具体信息\t\t\t|\n");printSomeSpace(space);

printf("|\t2.进行修改\t\t\t\t|\n");printSomeSpace(space);

printf("|\t3.进行删除\t\t\t\t|\n");printSomeSpace(space);

printf("|\t4.进行增添\t\t\t\t|\n");printSomeSpace(space);

printf("|\t5.快速查询\t\t\t\t|\n");printSomeSpace(space);

printf("|\t6.生成统计报表\t\t\t\t|\n");printSomeSpace(space);

printf("|\t7.返回上级菜单\t\t\t\t|\n");printSomeSpace(space);

printf("--------------按数字选择功能-----------------\n");printSomeSpace(space);

}

//rewrite file from list

void updateRoutesFILE(route\* routeHeadP) {

route \* routeP = routeHeadP;

FILE \*fRouteP = fopen("save/routes.txt", "w+");//从头改写文本

while (routeP != NULL) {

fprintf(fRouteP,"%s;%s;%d;%f;%f;%s;%s;%s;%s;%s;%s\n",routeP->routeID, routeP->name, routeP->siteNums, routeP->miles, routeP->period, routeP->startSite, routeP->endSite, routeP->adminName, routeP->tel, routeP->mTel, routeP->email);

routeP = routeP->next;

}

fclose(fRouteP);

}

void updateSitesFILE(site \* siteHeadP) {

site \* siteP = siteHeadP;

char \* url = (char\*)malloc(sizeof(char) \* 20);

strcpy(url, "save/");

strcat(url, siteHeadP->routeID);

strcat(url, ".txt");

FILE \*fSiteP = fopen(url, "w+");//从头改写文本

while (siteP != NULL) {

fprintf(fSiteP, "%s;%d;%s;%s;%f;%f;%f;%f;%s\n",siteP->routeID,siteP->siteSID, siteP->siteID, siteP->siteName, siteP->d2Start, siteP->d2Last, siteP->time2Last, siteP->waitTime, siteP->routeIDArray);

siteP = siteP->next;

}

fclose(fSiteP);

free(url);

}

void updateCarsFILE(car \* carHeadP, char \* fileName) {

car \* carP = carHeadP;

char \* url = (char\*)malloc(sizeof(char) \* 20);

strcpy(url, "save/");

strcat(url, fileName);//for example s00100

strcat(url, ".txt");

FILE \*fCarP = fopen(url, "w+");//从头改写文本

while (carP != NULL) {

fprintf(fCarP, "%s;%s;%s;%s\n", carP->carID, carP->routeID, carP->driverName, carP->driverTel);

carP = carP->next;

}

fclose(fCarP);

//Update good file automatically, 'cause it's too small to open an another method to realize it.

carP = carHeadP;//init again

while (carP != NULL) {

strcpy(url, "save/");

strcat(url, carP->carID);

strcat(url, ".txt");

FILE \* fGoodP = fopen(url, "w+");

good \* goodP = carP->good;

fprintf(fGoodP, "%s;%f;%s;%f\n", goodP->uploadType, goodP->upVolume, goodP->downloadType, goodP->downVolume);

carP = carP->next;

fclose(fGoodP);

}

free(url);

}

//The new route would be the last node in this list.

void addRoute(route \* routeHeadP) {

route \* newRouteP = AddRouteNode(routeHeadP, sizeRouteList(routeHeadP));

if (newRouteP != NULL) {

char inputTemp[51];

printSomeSpace(60);

printf("---------------请输入新路线的信息----------------\n");

printSomeSpace(60);printf("请输入编号:");

noNfgets(inputTemp, 50, stdin);

strcpy(newRouteP->routeID, inputTemp);

printSomeSpace(60);printf("请输入名称:");

noNfgets(inputTemp, 50, stdin);

strcpy(newRouteP->name, inputTemp);

newRouteP->firstSite = NULL;//因为是新的路线，故将指向的站点内容指针置空

newRouteP->siteNums = 0;//auto complete siteConut. default -> 0;

newRouteP->miles = 0;

newRouteP->period = 0;

printSomeSpace(60);printf("请输入起始站点编号:");

noNfgets(inputTemp, 50, stdin);

strcpy(newRouteP->startSite, inputTemp);

printSomeSpace(60);printf("请输入终止站点编号:");

noNfgets(inputTemp, 50, stdin);

strcpy(newRouteP->endSite, inputTemp);

printSomeSpace(60);printf("请输入负责人姓名:");

noNfgets(inputTemp, 50, stdin);

strcpy(newRouteP->adminName, inputTemp);

printSomeSpace(60);printf("请输入负责人办公室电话:");

noNfgets(inputTemp, 50, stdin);

strcpy(newRouteP->tel, inputTemp);

printSomeSpace(60);printf("请输入负责人移动电话:");

noNfgets(inputTemp, 50, stdin);

strcpy(newRouteP->mTel, inputTemp);

printSomeSpace(60);printf("请输入负责人电子邮箱:");

noNfgets(inputTemp, 50, stdin);

strcpy(newRouteP->email, inputTemp);

} else {

system("cls");

printf("addRoute error\n");

}

}

site\* addSite(site \* siteHeadP, char\* routeIDIN, route \* routeHeadP){

printf("---------------请输入新站点的信息----------------\n");

printSomeSpace(60);printf("请输入新站点在路线中的序号:");//新站点的位置

int seq = 0; scanf("%d%\*c", &seq);

int flag = 0;

site \* newSiteP = NULL;

if (siteHeadP == NULL){

newSiteP = (site\*)malloc(sizeof(site));

flag = 1;//the new head pointer( siteHeadP is a NULL pointer

} else {

newSiteP = AddSiteNode(siteHeadP, seq - 1);

}

if (newSiteP != NULL) {

newSiteP->carHeadP = NULL;

char inputTemp[51];

printSomeSpace(60);printf("请输入站点编号:");

noNfgets(inputTemp, 50, stdin);

strcpy(newSiteP->siteID, inputTemp);

//检测是否站点重复

route \* routeP = routeHeadP;

while(routeP != NULL){

site \* siteP = routeP->firstSite;

while(siteP != NULL){

if(!strcmp(siteP->siteID, newSiteP->siteID)){//一样

newSiteP->carHeadP = siteP->carHeadP;

}

siteP = siteP->next;

}

routeP = routeP->next;

}

printSomeSpace(60);printf("请输入站点名称:");

noNfgets(inputTemp, 50, stdin);

strcpy(newSiteP->siteName, inputTemp);

printSomeSpace(60);printf("请输入与上一个站点交通耗时:");

float fPeriod = 0;

scanf("%f%\*c", &fPeriod);

newSiteP->time2Last = fPeriod;

printSomeSpace(60);printf("请输入停留耗时:");

float waitTimeN = 0;

scanf("%f%\*c", &waitTimeN);

newSiteP->waitTime = waitTimeN;

printSomeSpace(60);printf("请输入经过本站点固定路线编号:");

noNfgets(inputTemp, 50, stdin);

strcpy(newSiteP->routeIDArray, inputTemp);

//总距离

if (seq == 1){

newSiteP->d2Start = 0;

newSiteP->d2Last = 0;

} else {

printSomeSpace(60);printf("请输入与上一个站点距离:");

float d2L = 0;

scanf("%f%\*c", &d2L);

newSiteP->d2Last = d2L;

site \* siteTempP = siteHeadP;

while(siteTempP->next != newSiteP){

siteTempP = siteTempP->next;

}

newSiteP->d2Start = siteTempP->d2Start + d2L;

}

//自动更新其他站点的离起始站点距离

updateAlld2S(newSiteP);

//自动补全序号

newSiteP->siteSID = seq;

strcpy(newSiteP->routeID, routeIDIN);

if (flag == 1){//the first one in new save file

newSiteP->next = NULL;

return newSiteP;

} else {

//自动更新被挤到后面的SID

site \* siteTemp = getSitePointer(siteHeadP, seq);

while (siteTemp != NULL) {

siteTemp->siteSID = siteTemp->siteSID + 1;

siteTemp = siteTemp->next;

}

return siteHeadP;

}

} else {

system("cls");

printSomeSpace(60);printf("addSite error\n");

return NULL;

}

}

//return the new car head pointer

car\* addCar(car \* carHeadP) {

int flag = 0;

car\* newCarP = NULL;

good\* newGoodP = NULL;

if (carHeadP == NULL){

newCarP = (car \*)malloc(sizeof(car));

flag = 1;

} else {//flag == 0

newCarP = AddCarNode(carHeadP, sizeCarList(carHeadP));

}

newGoodP = (good \*)malloc(sizeof(good));

newCarP->good = newGoodP;

char inputTemp[51];//auto free

printf("---------------请输入新路线的信息----------------\n");

printSomeSpace(60);printf("请输入车辆牌照:");

noNfgets(inputTemp, 50, stdin);

strcpy(newCarP->carID, inputTemp);

printSomeSpace(60);printf("请输入执行配送路线编号:");

noNfgets(inputTemp, 50, stdin);

strcpy(newCarP->routeID, inputTemp);

printSomeSpace(60);printf("请输入司机姓名:");

noNfgets(inputTemp, 50, stdin);

strcpy(newCarP->driverName, inputTemp);

printSomeSpace(60);printf("请输入司机移动电话:");

noNfgets(inputTemp, 50, stdin);

strcpy(newCarP->driverTel, inputTemp);

printSomeSpace(60);printf("请输入载货货物种类:");

noNfgets(inputTemp, 50, stdin);

strcpy(newCarP->good->uploadType, inputTemp);

printSomeSpace(60);printf("请输入载货货物容量:");

float upV = 0; scanf("%f%\*c", &upV);

newCarP->good->upVolume = upV;

printSomeSpace(60);printf("请输入卸货货物种类:");

noNfgets(inputTemp, 50, stdin);

strcpy(newCarP->good->downloadType, inputTemp);

printSomeSpace(60);printf("请输入卸货货物容量:");

float downV = 0; scanf("%f%\*c", &downV);

newCarP->good->downVolume = downV;

if (flag == 0){

return carHeadP;

} else {//flag == 1

newCarP->next = NULL;

return newCarP;

}

}

//不读取换行符的fgets，升级版

char\* noNfgets(char \* Buffer, int MaxConut, FILE\* Stream) {

char\* returnPointer = fgets(Buffer, MaxConut, Stream);//now there is data in Buffer with '\n'.

register int i = 0;

while (Buffer[i] != '\n') {

i++;

}

//now i points '\n'

Buffer[i] = '\0';

return returnPointer;

}

//修改信息逻辑

int changeRoute(route \* routeSpecific) {

printSomeSpace(60);

printf("-------------------------------------------------\n");

char temp[] = " ";

printf("%60s1.编号\n%60s2.名称\n%60s3.总公里数\n%60s4.起始站点编号\n%60s5.终止站点编号\n%60s6.负责人姓名\n%60s7.负责人办公室电话\n%60s8.负责人移动电话\n%60s8.负责人电子邮箱\n\n%60s请选择你要修改的属性:",temp,temp,temp,temp,temp,temp,temp,temp,temp,temp,temp);

int choose = 0;

scanf("%d%\*c", &choose);

char input[51];//no free

switch (choose) {

case 1:

printSomeSpace(60);printf("请输入新的编号:");

char \* oldID = (char \*)malloc(sizeof(char) \* 30);

strcpy(oldID, routeSpecific->routeID);

noNfgets(input, 50, stdin);

strcpy(routeSpecific->routeID, input);

//改变其编号对应的存档文件名

changeSaveName(oldID, routeSpecific->routeID);

changeOldRouteID(routeSpecific);

free(oldID);

break;

case 2:

printSomeSpace(60);printf("请输入新的名称:");

noNfgets(input, 50, stdin);

strcpy(routeSpecific->name, input);

break;

case 3:

printSomeSpace(60);printf("请输入新的总公里数:");

float newMiles = 0;

scanf("%f%\*c", &newMiles);

routeSpecific->miles = newMiles;

break;

routeSpecific->period = 0;

case 4:

printSomeSpace(60);printf("请输入新的起始站点编号:");

noNfgets(input, 50, stdin);

strcpy(routeSpecific->startSite, input);

break;

case 5:

printSomeSpace(60);printf("请输入新的终止站点编号:");

noNfgets(input, 50, stdin);

strcpy(routeSpecific->endSite, input);

break;

case 6:

printSomeSpace(60);printf("请输入新的负责人姓名:");

noNfgets(input, 50, stdin);

strcpy(routeSpecific->adminName, input);

break;

case 7:

printSomeSpace(60);printf("请输入新的负责人办公室电话:");

noNfgets(input, 50, stdin);

strcpy(routeSpecific->tel, input);

break;

case 8:

printSomeSpace(60);printf("请输入新的负责人移动电话:");

noNfgets(input, 50, stdin);

strcpy(routeSpecific->mTel, input);

break;

case 9:

printSomeSpace(60);printf("请输入新的负责人电子邮箱:");

noNfgets(input, 50, stdin);

strcpy(routeSpecific->email, input);

break;

default:

printf("输入有误\n");

return 0;

}

return 1;

}

int changeSite(site \* siteSpecific, int notFirst, site \* siteHeadP) {

printSomeSpace(60);

printf("----------------------------------------------\n");

char space[] = " ";

if (notFirst){

printf("%60s1.站点编号\n%60s2.站点名称\n%60s3.经过本站点的固定路线编号\n%60s4.停留耗时\n%60s5.与上一站交通耗时\n%60s6.与上一个站点距离\n\n%60s请选择你要修改的属性:",space,space,space,space,space,space,space);

} else {

printf("%60s1.站点编号\n%60s2.站点名称\n%60s3.经过本站点的固定路线编号\n%60s4.停留耗时\n%60s5.与上一站交通耗时\n\n%60s请选择你要修改的属性:",space,space,space,space,space,space);

}

int choose = 0;

scanf("%d%\*c", &choose);

char input[50];//auto free

printSomeSpace(60);

switch (choose) {

case 1:

printf("请输入新的站点编号:");

char \* oldID = (char \*)malloc(sizeof(char) \* 30);

strcpy(oldID, siteSpecific->siteID);

noNfgets(input, 50, stdin);

strcpy(siteSpecific->siteID, input);//向链表内部更新新的编号

changeSaveName(oldID, siteSpecific->siteID);//改变其编号对应的存档文件名（联动改变下文

free(oldID);

return 1;

case 2:

printf("请输入新的站点名称:");

noNfgets(input, 50, stdin);

strcpy(siteSpecific->siteName, input);

return 2;

case 3:

printf("请输入新的经过本站点的固定路线编号:");

noNfgets(input, 50, stdin);

strcpy(siteSpecific->routeIDArray, input);

return 3;

case 4:

printf("请输入新的停留耗时:");

float newPeriod = 0;

scanf("%f%\*c", &newPeriod);

siteSpecific->waitTime = newPeriod;

return 4;

case 5:

printf("请输入新的与上一站耗时:");

newPeriod = 0;

scanf("%f%\*c", &newPeriod);

siteSpecific->time2Last = newPeriod;

return 5;

case 6:

if (!notFirst){

printSomeSpace(60);printf("输入有误!\n");

return 0;

}

printf("请输入新的与上一个站点距离:");

float d2LastNew = 0;

scanf("%f%\*c", &d2LastNew);

siteSpecific->d2Last = d2LastNew;

site \* siteCorrectP = siteHeadP;

while (siteCorrectP->next != siteSpecific){

siteCorrectP = siteCorrectP->next;

}

updateAlld2S(siteCorrectP);

return 6;

default:

printSomeSpace(60);printf("输入有误!\n");

return 0;

}

}

int changeCar(car \* carSpecific) {

printSomeSpace(60);printf("-------------------------------------------------\n");

char space[] = " ";

printf("%60s1.车辆牌照\n%60s2.司机姓名\n%60s3.司机移动电话\n%60s4.载货货物种类\n%60s5.载货货物容量\n%60s6.卸货货物种类\n%60s7.卸货货物容量\n\n%60s请选择你要修改的属性:",space,space,space,space,space,space,space,space);

int choose = 0;

scanf("%d%\*c", &choose);

char input[50];//auto free

printSomeSpace(60);

switch (choose) {

case 1:

printf("请输入新的车辆牌照:");

char \* oldID = (char \*)malloc(sizeof(char) \* 30);

strcpy(oldID, carSpecific->carID);

noNfgets(input, 50, stdin);

strcpy(carSpecific->carID, input);//向链表内部更新新的编号

changeSaveName(oldID, carSpecific->carID);//改变其编号对应的存档文件名（联动改变下文

free(oldID);

break;

case 2:

printf("请输入新的司机姓名:");

noNfgets(input, 50, stdin);

strcpy(carSpecific->driverName, input);

break;

case 3:

printf("请输入新的司机移动电话:");

noNfgets(input, 50, stdin);

strcpy(carSpecific->driverTel, input);

break;

case 4:

printf("请输入新的载货货物种类:");

noNfgets(input, 50, stdin);

strcpy(carSpecific->good->uploadType, input);

break;

case 5:

printf("请输入新的载货货物容量:");

float newUpV = 0;

scanf("%f%\*c", &newUpV);

carSpecific->good->upVolume = newUpV;

break;

case 6:

printf("请输入新的卸货货物种类:");

noNfgets(input, 50, stdin);

strcpy(carSpecific->good->downloadType, input);

break;

case 7:

printf("请输入新的卸货货物容量:");

float newDownV = 0;

scanf("%f%\*c", &newDownV);

carSpecific->good->downVolume = newDownV;

break;

default:

printf("输入有误!\n");

return 0;

}

return 1;

}

//输出站点信息逻辑

void printSitePage(route \*routeP, route \* routeHeadP) {

int inSitePage = 1;

site\* siteHeadP = routeP->firstSite;

while (inSitePage) {

system("cls");

site\* siteP = siteHeadP;

system("date /T");

printSomeG(71);printf("路线编号：%6s 的所有站点信息", routeP->routeID);printSomeG(71);printf("\n");//title

char second[] = "序号";

char third[] = "站点编号";

char fourth[] = "站点名称";

char fifth[] = "与起始站距离";

char sixth[] = "与上一个站距离";

char seventh[] = "与上一站交通耗时";

char eighth[] = "停留耗时";

char ninth[] = "经过本站点的路线编号";

printf("|%-4s %-10s %-50s %-12s %-14s %-16s %-8s %-50s|\n", second, third, fourth, fifth, sixth, seventh, eighth, ninth);

printf("|");printSomeSpace(171);printf("|\n");

while (siteP != NULL) {

printf("|%-4d %-10s %-50s %-12.3f %-14.3f %-16.3f %-8.3f %-50s|\n", siteP->siteSID, siteP->siteID, siteP->siteName, siteP->d2Start, siteP->d2Last, siteP->time2Last, siteP->waitTime, siteP->routeIDArray);

siteP = siteP->next;

}

printSomeG(173);printf("\n");

printSomeSpace(60);printf("|\t1.查看站点具体信息\t\t\t|\n");

printSomeSpace(60);printf("|\t2.进行修改\t\t\t\t|\n");

printSomeSpace(60);printf("|\t3.进行删除\t\t\t\t|\n");

printSomeSpace(60);printf("|\t4.进行增添\t\t\t\t|\n");

printSomeSpace(60);printf("|\t5.返回上级菜单\t\t\t\t|\n");

printSomeSpace(60);printf("--------------按数字选择功能-----------------\n");

int seq = 0;

printSomeSpace(60);scanf("%d%\*c", &seq);

printSomeSpace(60);

switch (seq) {

case 1:

printf("想查看第几个站点的详细车辆信息：");

scanf("%d%\*c", &seq);

printCarPage(getSitePointer(siteHeadP, seq - 1));//进去该函数

break;

case 2:

printf("想修改第几个站点的信息:");

scanf("%d%\*c", &seq);

int changeWhich = changeSite(getSitePointer(siteHeadP, seq - 1), seq-1, siteHeadP);

updateSitesFILE(siteHeadP);

MessageBox(NULL, TEXT("成功修改站点信息"), TEXT("操作成功"), MB\_OK);

if (changeWhich == 6){//更改了距离，应更改路线内的总里程和所有被修改站点以后（包括自身）的与起始站点的距离

routeP->miles = totalMile(routeP);

} else if (changeWhich == 4 || changeWhich == 5){//更改了时间相关

routeP->period = totalTime(routeP);

} else {

break;

}

updateRoutesFILE(routeHeadP);//update save file

break;

case 3:

printf("想删除第几个站点的信息:");

scanf("%d%\*c", &seq);

siteHeadP = DelSitePos(siteHeadP, seq - 1);//get the new site head pointer

routeP->firstSite = siteHeadP;

//自动改变上文

if (seq == 1) {//删除了第一个

strcpy(routeP->startSite, siteHeadP->siteID);//更新上一级存储的第一站点名称数据

} else if (seq == (sizeSiteList(siteHeadP)-2)) {//删除了最后一个

strcpy(routeP->endSite, getSitePointer(siteHeadP, sizeSiteList(siteHeadP) - 1)->siteID);//更新上一级存储的终点站点名称数据

}

//自动更新序号

site \* stepSite = getSitePointer(siteHeadP, seq - 1);

while (stepSite != NULL) {

stepSite->siteSID = stepSite->siteSID - 1;

stepSite = stepSite->next;

}

updateSitesFILE(siteHeadP);

//更新路线站点数信息至链表与存档

updateSitesCount(routeP);

routeP->period = totalTime(routeP);

//更新路线距离

routeP->miles = totalMile(routeP);

//更新D2S

updateAlld2S(siteHeadP);

updateRoutesFILE(routeHeadP);//update save file

//TODO: 不方便删除站点数据，应筛选是否有被其他路线占用

MessageBox(NULL, TEXT("成功删除站点"), TEXT("操作成功"), MB\_OK);

break;

case 4:

//增添逻辑

siteHeadP = addSite(siteHeadP, routeP->routeID, routeHeadP);

routeP->firstSite = siteHeadP;

updateSitesFILE(siteHeadP);

//更新路线站点数信息至链表与存档

updateSitesCount(routeP);

routeP->period = totalTime(routeP);

//更新路线距离

routeP->miles = totalMile(routeP);

//更新上级路线的起始站点末尾站点信息

site \* siteTempP = siteHeadP;

strcpy(routeP->startSite, siteHeadP->siteID);

while(siteTempP->next != NULL){

siteTempP = siteTempP->next;

}

strcpy(routeP->endSite, siteTempP->routeID);

updateRoutesFILE(routeHeadP);

MessageBox(NULL, TEXT("成功增添站点"), TEXT("操作成功"), MB\_OK);

break;

case 5:

inSitePage = 0;

break;

default:

printf("输入错误！");//输入的数字选项错误

break;

}

system("cls");

}

}

//输出车辆信息

void printCarPage(site \* siteP) {

car \* carHeadP = siteP->carHeadP;

int inCarPage = 1;

while (inCarPage) {

system("cls");

car \* carP = carHeadP;

system("date /T");

printSomeG(65);printf("站点编号：%6s 的所有车辆信息", siteP->siteID);printSomeG(50);printf("\n");

char first[] = "车辆牌照";

char second[] = "执行路线编号";

char third[] = "司机姓名";

char fourth[] = "司机移动电话";

char fifth[] = "载货种类";

char sixth[] = "载货容量";

char seventh[] = "卸货种类";

char eighth[] = "卸货容量";

printf("|%-8s %-12s %-8s %-12s %-10s %-10s %-10s %-10s", first, second, third, fourth, fifth, sixth, seventh, eighth);printSomeSpace(57);

printf("|\n|");printSomeSpace(144);printf("|\n");

while (carP != NULL) {

printf("|%-8s %-12s %-8s %-12s ", carP->carID, carP->routeID, carP->driverName, carP->driverTel);

if (carP->good != NULL){

printf("%-10s %-10.3f %-10s %-10.3f", carP->good->uploadType, carP->good->upVolume, carP->good->downloadType, carP->good->downVolume);

}

printSomeSpace(57);

printf("|\n");

carP = carP->next;

}//now carP == NULL

printSomeG(146);printf("\n");

printSomeSpace(60);printf("|\t1.进行修改\t\t\t\t|\n");

printSomeSpace(60);printf("|\t2.进行删除\t\t\t\t|\n");

printSomeSpace(60);printf("|\t3.进行增添\t\t\t\t|\n");

printSomeSpace(60);printf("|\t4.返回上级菜单\t\t\t\t|\n");

printSomeSpace(60);printf("--------------按数字选择功能-----------------\n");

printSomeSpace(60);

int seq = 0; scanf("%d%\*c", &seq);

printSomeSpace(60);

switch (seq) {

case 1:

printf("想修改第几台车辆的信息:");

scanf("%d%\*c", &seq);

changeCar(getCarPointer(carHeadP, seq - 1));

updateCarsFILE(carHeadP, siteP->siteID);

MessageBox(NULL, TEXT("成功修改车辆信息"), TEXT("操作成功"), MB\_OK);

break;

case 2:

printf("想删除第几台车辆的信息:");

scanf("%d%\*c", &seq);

//删除对应的存档文件

char \*url = (char \*)malloc(sizeof(char) \* 20);

strcpy(url, "save/");

car \* carDel = getCarPointer(carHeadP, seq - 1);

if (carDel != NULL){

strcat(url, getCarPointer(carHeadP, seq - 1)->carID);

strcat(url, ".txt");

remove(url);

} else {

printf("输入指令错误\n");

}

free(url);

carHeadP = DelCarPos(carHeadP, seq - 1);

updateCarsFILE(carHeadP, siteP->siteID);

MessageBox(NULL, TEXT("成功删除车辆"), TEXT("操作成功"), MB\_OK);

break;

case 3:

carHeadP = addCar(carHeadP);

siteP->carHeadP = carHeadP;

updateCarsFILE(carHeadP, siteP->siteID);

MessageBox(NULL, TEXT("成功增添车辆"), TEXT("操作成功"), MB\_OK);

break;

case 4:

inCarPage = 0;

break;

default:

break;

}

}

}

//change the save file's name

int changeSaveName(char\* old, char\* new) {

char \* oldUrl = (char\*)malloc(20 \* sizeof(char));

char \* newUrl = (char\*)malloc(20 \* sizeof(char));

strcpy(oldUrl, "save/");

strcat(oldUrl, old);

strcat(oldUrl, ".txt");

strcpy(newUrl, "save/");

strcat(newUrl, new);

strcat(newUrl, ".txt");

int result = 0;

if (rename(oldUrl, newUrl) == 0) {

result = 1;

} else {

result = 0;

}

free(oldUrl);

free(newUrl);

return result;

}

//用来改变存储某路线所有站点的存档中的routeID

void changeOldRouteID(route \* routeSpecific) {

site \* siteP = routeSpecific->firstSite;

while (siteP != NULL) {

strcpy(siteP->routeID, routeSpecific->routeID);

siteP = siteP->next;

}

updateSitesFILE(routeSpecific->firstSite);

}

//自动站点数更新（根据二级指针的的返回值，为一级的成员数据“站点数”进行赋值）

void updateSitesCount(route \* routeSpecific){

int count = 0;

site \* siteHeadP = routeSpecific->firstSite;

count = sizeSiteList(siteHeadP);

routeSpecific->siteNums = count;

}

//生成统计报表

void makeCsv(route \* routeHeadP){

system("cls");

FILE \* saveF = fopen("物流信息统计报表.csv","w");

fprintf(saveF, "所有路线信息：\n");

fprintf(saveF, "·,编号,名称,总站点数,总公里数,总耗时,起始站点编号,终止站点编号,负责人姓名,负责人办公室电话,负责人移动电话,负责人电子邮箱\n");

route \* routeP = routeHeadP;

while (routeP != NULL){

fprintf(saveF,"·,%s,%s,%d,%f,%f,%s,%s,%s,%s,%s,%s\n",routeP->routeID, routeP->name, routeP->siteNums, routeP->miles, routeP->period, routeP->startSite, routeP->endSite, routeP->adminName, routeP->tel, routeP->mTel, routeP->email);

routeP = routeP->next;

}

fprintf(saveF, "\n\n所有站点信息：\n");

routeP = routeHeadP;

while (routeP != NULL){

site \* siteP = routeP->firstSite;

fprintf(saveF, "\n·,路线%s 的站点信息：\n", routeP->routeID);

fprintf(saveF, "·,固定配送路线编号,序号,编号,名称,与起始站点距离,与上一个站点距离,与上一个站点交通耗时,停留耗时,经过本站点固定路线编号\n");

while (siteP != NULL){

fprintf(saveF, "·,%s,%d,%s,%s,%f,%f,%f,%f,%s\n",siteP->routeID,siteP->siteSID, siteP->siteID, siteP->siteName, siteP->d2Start, siteP->d2Last, siteP->time2Last, siteP->waitTime, siteP->routeIDArray);

siteP = siteP->next;

}

routeP = routeP->next;

}

fprintf(saveF, "\n\n所有车辆信息：\n");

routeP = routeHeadP;

while (routeP != NULL){

site \* siteP = routeP->firstSite;

while (siteP != NULL){

car \* carP = siteP->carHeadP;

fprintf(saveF, "\n·,站点%s 的车辆信息：\n", siteP->siteID);

fprintf(saveF, "·,车辆牌照,执行配送路线编号,司机姓名,司机移动电话,载货种类,载货容量,卸货种类,卸货容量\n");

while (carP != NULL){

fprintf(saveF, "·,%s,%s,%s,%s", carP->carID, carP->routeID, carP->driverName, carP->driverTel);

good \* goodP = carP->good;

if (goodP != NULL){

fprintf(saveF, "·,%s,%f,%s,%f", goodP->uploadType, goodP->upVolume, goodP->downloadType, goodP->downVolume);

}

fprintf(saveF, "\n");

carP = carP->next;

}

siteP = siteP->next;

}

routeP = routeP->next;

}

fprintf(saveF, "\n\n统计信息：\n\n");

fprintf(saveF, "·,路线总数：,%d\n",sizeRouteList(routeHeadP));

int sitesCount = 0;routeP = routeHeadP;

while(routeP != NULL){

sitesCount = sitesCount + sizeSiteList(routeP->firstSite);

routeP = routeP->next;

}

fprintf(saveF, "·,站点总数：,%d\n", sitesCount);

int carsCount = 0;routeP = routeHeadP;

while (routeP != NULL){

site \* siteP = routeP->firstSite;

while (siteP != NULL){

carsCount = carsCount + sizeCarList(siteP->carHeadP);

siteP = siteP->next;

}

routeP = routeP->next;

}

fprintf(saveF, "·,车辆总数：,%d\n\n", carsCount);

//计算里程逻辑

float mileSum = 0;routeP = routeHeadP;

while (routeP != NULL){

mileSum = mileSum + routeP->miles;

routeP = routeP->next;

}

fprintf(saveF, "·,路线总里程：,%.4f\n", mileSum);

fprintf(saveF, "·,路线平均里程：,%.4f\n", mileSum/sizeRouteList(routeHeadP));

char StringTemp[50];

float tempFloat = -1;

routeP = routeHeadP;//比较之前的初始化

while (routeP != NULL){

if (routeP->period > tempFloat){

tempFloat = routeP->miles;

strcpy(StringTemp, routeP->routeID);

}

routeP = routeP->next;

}

fprintf(saveF, "·,最长里程数：,%.4f,路线%s\n", tempFloat, StringTemp);

tempFloat = 99999;routeP = routeHeadP;

while (routeP != NULL){

if (routeP->period < tempFloat){

tempFloat = routeP->miles;

strcpy(StringTemp, routeP->routeID);

}

routeP = routeP->next;

}

fprintf(saveF, "·,最短里程数：,%.4f,路线%s\n\n", tempFloat, StringTemp);

//计算耗时逻辑

tempFloat = -1;routeP = routeHeadP;

while (routeP != NULL){

float timeThis = totalTime(routeP);

if (timeThis > tempFloat){

tempFloat = timeThis;

strcpy(StringTemp, routeP->routeID);

}

routeP = routeP->next;

}

fprintf(saveF, "·,最长耗时：,%.4f,路线%s\n", tempFloat, StringTemp);

tempFloat = 99999;routeP = routeHeadP;

while (routeP != NULL){

float timeThis = totalTime(routeP);

if (timeThis < tempFloat){

tempFloat = timeThis;

strcpy(StringTemp, routeP->routeID);

}

routeP = routeP->next;

}

fprintf(saveF, "·,最短耗时：,%.4f,路线%s\n", tempFloat, StringTemp);

tempFloat = 0;routeP = routeHeadP;

while (routeP != NULL){

tempFloat = tempFloat + totalTime(routeP);

routeP = routeP->next;

}

fprintf(saveF, "·,平均耗时：,%.4f\n\n", tempFloat/sizeRouteList(routeHeadP));

//统计载货量逻辑

tempFloat = 0;routeP = routeHeadP;

float minCarry = 9999;float maxCarry = -1;

float minEfficiency = 999;float maxEfficiency = -1;

char maxCarID[10];char minCarID[10];

char maxCarID2[10];char minCarID2 [10];

while (routeP != NULL){

site \* siteP = routeP->firstSite;

while (siteP != NULL){

car \* carP = siteP->carHeadP;

while (carP != NULL){

if (carP->good != NULL){

float carryVolume = carP->good->upVolume;

tempFloat = tempFloat + carryVolume;

//更新最大最小载货量数据

if (carryVolume > maxCarry){

maxCarry = carryVolume;

strcpy(maxCarID, carP->carID);

} else if (carryVolume < minCarry){

minCarry = carryVolume;

strcpy(minCarID, carP->carID);

}

//更新最高最低效率数据

float efficiencyThis = carryVolume/getMile(carP->routeID, routeHeadP);

if (efficiencyThis > maxEfficiency){

maxEfficiency = efficiencyThis;

strcpy(maxCarID2, carP->carID);

} else if (efficiencyThis < minEfficiency){

minEfficiency = efficiencyThis;

strcpy(minCarID2, carP->carID);

}

}

carP = carP->next;

}

siteP = siteP->next;

}

routeP = routeP->next;

}

fprintf(saveF, "·,总载货量：,%.4f\n", tempFloat);

fprintf(saveF, "·,平均载货量：,%.4f\n", tempFloat/carsCount);

fprintf(saveF, "·,最大载货量：,%.4f,%s\n", maxCarry, maxCarID);

fprintf(saveF, "·,最小载货量：,%.4f,%s\n\n", minCarry, minCarID);

//计算配送效率逻辑

fprintf(saveF, "·,平均配送效率：,%.4f\n", tempFloat/mileSum);

fprintf(saveF, "·,最高配送效率：,%.4f,%s\n", maxEfficiency, maxCarID2);

fprintf(saveF, "·,最低配送效率：,%.4f,%s\n", minEfficiency, minCarID2);

fprintf(saveF, "·,PS:配送效率指载货量与配送时间之比\n");

fclose(saveF);

MessageBox(NULL, TEXT("报表已经保存在程序的根目录。"), TEXT("操作成功"), MB\_OK);

}

//从传入的参数对应的路线中，进入站点数据中去计算准确的耗时数据

float totalTime(route \*routeP){

site \* siteP = routeP->firstSite;

float time = 0;

while (siteP != NULL){

time = siteP->d2Last + siteP->waitTime + time;

siteP = siteP->next;

}

return time;

}

//从传入的参数对应的路线中，进入站点数据中去获得总长度

float totalMile(route \*routeP){

site \* siteTempP = routeP->firstSite;

while (siteTempP->next != NULL){

siteTempP = siteTempP->next;

}

return siteTempP->d2Start;

}

//从传入的路线链表中寻找对应字符串参数的数据的里程

float getMile(char \*routeIDThis, route \* routeHeadP){

route \*routeP = routeHeadP;

while (routeP != NULL){

if (strcmp(routeP->routeID, routeIDThis) == 0){

return routeP->miles;

}

routeP = routeP->next;

}

return 0;

}

//批量输出空格

void printSomeSpace(int num){

int i = 0;

for (i = 0; i < num; i++){

printf(" ");

}

}

//批量输出Gang 杠号

void printSomeG(int num){

int i = 0;

for (i = 0; i < num; i++){

printf("-");

}

}

/\*\*

\* 更新路线的d2S

\* 传入的站点指针为数据正确的最后一个路线，往后的所有路线都需要进行更新

\*/

void updateAlld2S(site \* siteP){//

site \* sitePre = siteP;

site \* siteNext = siteP->next;

while (siteNext != NULL){

siteNext->d2Start = sitePre->d2Start + siteNext->d2Last;

sitePre = siteNext;

siteNext = siteNext->next;

}

}