****

**J I A N G S U U N I V E R S I T Y**

**数据结构与算法课程设计**

|  |  |
| --- | --- |
| 所属学院： | 计算机科学与通信技术学院 |
| 专业班级： | 物联网2102 |
| 姓 名： | 孟德昊 |
| 学 号： | 3201303014 |
| 指导教师： | 郑文怡 |

2022年6月

目录

[一、问题分析和任务定义 1](#_Toc107517338)

[1.1题目简述 1](#_Toc107517339)

[1.2功能模块要求 1](#_Toc107517340)

[二、数据结构的选择和概要设计 1](#_Toc107517341)

[2.1头文件声明与宏定义 1](#_Toc107517342)

[2.2结构体定义 3](#_Toc107517343)

[三、详细设计和编码 3](#_Toc107517344)

[3.1初始化系统 3](#_Toc107517345)

[3.2菜单选择 8](#_Toc107517346)

[四、运行结果及数据测试 26](#_Toc107517347)

[五、小结 29](#_Toc107517348)

[六、源码 31](#_Toc107517349)

[Menu.h 31](#_Toc107517350)

[Menu.cpp 32](#_Toc107517351)

[Main.cpp 55](#_Toc107517352)

[WalkRoutes.txt 55](#_Toc107517353)

## 一、问题分析和任务定义

### 1.1题目简述

为2022级新生定制一个线路问询交互系统，依据参考因素，例如：换乘路线的路径最短、耗费时间最短、所需车资最少等，经过分析处理得到可达目的站点换乘次数最少的乘车方案，具体可分为：

(1)零次换乘 起始站点和目的站点之间存在可直达的公交线路，即出行无需转乘就可以直接到达目的站点，这也是较为理想的方案。

(2)一次换乘 起始站点和目的站点之间没有公交车直接往返，即两站点之间不存在可直达的公交线路，则出行居民需要在途经的某个站点下车，然后转乘另一线路公交车才能达到目的站点。

(3)步行 基于地图用户可以按照最近距离步行前往某个地点。

### 1.2功能模块要求

本次课程需要以校内公交线路为例，要求本系统能提供一个友好的功能菜单，菜单中应至少包括（但不限于）以下几个基本功能：

1) 初始化系统：

基本信息初始化，包括公交线路编号、站点名称、站点之间票价、耗费时间、发车时间等（根据自己设计可以增加相关公交车信息，）；

基本要求：将公交线路的基本信息存于文件，每次运行程序时，通过文件对基本信息初始化；

2) 公交信息维护：

由于修路等城市规划变动的原因，公交线路可能会有微调，例如需要增加、删除或者修改部分站点的信息；修改之后的课程信息写回文件，便于下一次初始化；

3) 查询功能：

从键盘输入编号，给出具体行车路线、行车时间等；

从键盘输入始发站和终点站，完成公交线路换乘查询（零次，1次）

最短线路规划：从五棵松至校内其余地点的最短路线。

4）用户角色：

分为管理员和学生，管理员可以进行初始化、公交信息维护和查询功能；游客只可以使用查询功能。

## 二、数据结构的选择和概要设计

### 2.1头文件声明与宏定义

#ifndef COURSEDESIGN\_MENU\_H

#define COURSEDESIGN\_MENU\_H

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <typeinfo>

using namespace std;

#define INFINITY 999999

class Menu {

private:

// 数据结构

typedef struct{

string StationName[10];

}Station;

typedef struct{

int StationNum; // 站台数目

Station StationName;

double money;

}Bus;

typedef struct{

string ExistStation[20]; // 站点名称

int size=0; // 已有站点的数量

}ExistStation;

// 图

typedef struct{

int numVertexes;

int numEdges;

int arc[20][20];

}MGraph;

ExistStation existStation;

MGraph WalkGraph,BusGraph;

vector<pair<string,Bus>> BusMessage; // 公交车信息

string UserName; // 用户名

string Password;// 密码

/\*方法\*/

vector<string> split(const string &s, const string &seperator);

bool Init();

bool LoginAdmin();

bool VerifyIdentidy();

bool AdminMenu();

bool MaintainMenu();

bool AddBusRoute();

bool PrintRoute();

bool PrintStation();

bool RoutExist(string BusName);

bool StationExist(string StationName);

bool DeleteRoute();

bool ChangeBusRoute();

bool DeleteBusStation();

bool AddBusStation();

bool StudentMenu();

bool FindRouteMenu();

bool FindStation();

bool RoutePlanMenu();

bool WalkMost(int SetoutNum,int GotoNum);

bool TimingMost(int SetoutNum,int GotoNum);

bool FiveTreeRoute();

bool Transfer(int SetOutNum,int GotoNUm);

bool Write();

bool HuanRoute();

public:

Menu();

};

#endif //COURSEDESIGN\_MENU\_H

### 2.2结构体定义

s typedef struct{

string StationName[10];

}Station;

typedef struct{

int StationNum; // 站台数目

Station StationName;

double money;

}Bus;

typedef struct{

string ExistStation[20]; // 站点名称

int size=0; // 已有站点的数量

}ExistStation;

// 图

typedef struct{

int numVertexes;

int numEdges;

int arc[20][20];

}MGraph;

## 三、详细设计和编码

### 3.1初始化系统

#### 3.1.1 数据初始化与保存更改站点信息

创建一个WalkRoutes.txt文件，通过下面部分代码读取或存储信息，完成初始化。

// 读文件时数据处理

vector<string> Menu::split(const string &s, const string &seperator) {

vector<string> result;

int i = 0;

while (i != s.size()) {

//找到字符串中首个不等于分隔符的字母；

int flag = 0;

while (i != s.size() && flag == 0) {

flag = 1;

for (int x = 0; x < seperator.size(); ++x) {

if (s[i] == seperator[x]) {

++i;

flag = 0;

break;

}

}

}

flag = 0;

int j = i;

while (j != s.size() && flag == 0) {

for (int x = 0; x < seperator.size(); ++x) {

if (s[j] == seperator[x]) {

flag = 1;

break;

}

}

if (flag == 0) ++j;

}

if (i != j) {

result.push\_back(string(s.substr(i, j - i)));

i = j;

}

}

return result;

}

/\* 读文件初始化 \*/

bool Menu::Init() {

ifstream ifs;

vector<string> tokens;

tokens.clear();

existStation.size = 0;

BusGraph.numEdges = 0;

BusGraph.numVertexes = 0;

WalkGraph.numVertexes = 0;

WalkGraph.numEdges = 0;

ifs.open("E:\\ujs\_meng\\Shuju\_struct\_c++\\CourseDesign\\WalkRoutes.txt", ios::in);

string s;

// 添加所有站点

getline(ifs, s);

do {

tokens.push\_back(s);

getline(ifs, s);

} while (s != "//");

tokens = split(tokens[0], ",");

for (auto item: tokens) {

vector<string> test = split(item, " ");

existStation.ExistStation[existStation.size++] = test[1];

}

// 输出所有站点

// for (int i = 0; i < existStation.size; ++i) {

// cout << existStation.ExistStation[i] << " ";

// }

cout << endl;

// 添加公交线路信息

getline(ifs, s);

do {

tokens.clear();

tokens.push\_back(s);

getline(ifs, s);

vector<string> test = split(tokens[0], ":");

string buf = test[0];

Bus buff;

buff.money = stof(test[2]);

buff.StationNum = test[1].size() / 2 + 1;

int j = 0;

vector<string> in = split(test[1], ",");

int i = 0;

for (auto item: in) {

buff.StationName.StationName[i++] = existStation.ExistStation[stoi(item)];

}

BusMessage.push\_back(make\_pair(buf, buff));

} while (s != "//");

// 输出公交线路信息

// for (auto item:BusMessage) {

// cout << item.first << " ";

// for (int i = 0; i < item.second.StationNum; ++i) {

// cout << item.second.StationName.StationName[i] << " ";

// }

// cout << " 票价为: " << item.second.money << "元";

// cout << endl;

// }

// 添加公交邻接矩阵

BusGraph.numVertexes = existStation.size;

for (int i = 0; i < BusGraph.numVertexes; ++i) {

for (int j = 0; j < BusGraph.numVertexes; ++j) {

BusGraph.arc[i][j] = BusGraph.arc[j][i] = INFINITY;

if (i == j) {

BusGraph.arc[i][j] = BusGraph.arc[j][i] = 0;

}

}

}

getline(ifs, s);

do {

tokens.clear();

tokens = split(s, ",");

BusGraph.arc[stoi(tokens[1])][stoi(tokens[0])] = BusGraph.arc[stoi(tokens[0])][stoi(tokens[1])] = stoi(

tokens[2]);

BusGraph.numEdges++;

getline(ifs, s);

} while (s != "//");

// 输出公交邻接矩阵

// cout << "输出公交邻接矩阵" << endl;

// for (int i = 0; i < BusGraph.numVertexes; ++i) {

// for (int j = 0; j < BusGraph.numVertexes; ++j) {

// cout << BusGraph.arc[i][j] << " ";

// }

// cout << endl;

// }

// 添加步行邻接矩阵

WalkGraph.numVertexes = existStation.size;

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

WalkGraph.arc[i][j] = WalkGraph.arc[j][i] = INFINITY;

if (i == j) {

WalkGraph.arc[i][j] = WalkGraph.arc[j][i] = 0;

}

}

}

getline(ifs, s);

do {

tokens.clear();

tokens = split(s, ",");

WalkGraph.arc[stoi(tokens[0])][stoi(tokens[1])] = WalkGraph.arc[stoi(tokens[1])][stoi(tokens[0])] = stoi(

tokens[2]);

WalkGraph.numEdges++;

} while (getline(ifs, s));

// 输出步行邻接矩阵

// cout << "输出步行邻接矩阵" << endl;

// for (int i = 0; i < WalkGraph.numVertexes; ++i) {

// for (int j = 0; j < WalkGraph.numVertexes; ++j) {

// cout << WalkGraph.arc[i][j] << " ";

// }

// cout << endl;

// }

ifs.close();

return true;

}

// 信息保存

bool Menu::Write() {

ofstream ofs;

ofs.open("E:\\ujs\_meng\\Shuju\_struct\_c++\\CourseDesign\\NewRoutes.txt",ios::out);

for (int i = 0; i < existStation.size-1; ++i) {

ofs << i << " " << existStation.ExistStation[i] << ",";

}

ofs << existStation.size << existStation.ExistStation[existStation.size] << endl;

ofs << "//" << endl;

for (int i = 0; i < BusGraph.numVertexes; ++i) {

for (int j = 0; j < BusGraph.numVertexes; ++j) {

if (BusGraph.arc[i][j] != INFINITY && i != j){

ofs << i << "," << j << "," << BusGraph.arc[i][j] << endl;

}

}

}

ofs << "//" << endl;

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

if (WalkGraph.arc[i][j] != INFINITY && i != j){

ofs << i << "," << j << "," << WalkGraph.arc[i][j] << endl;

}

}

}

ofs.close();

return true;

}

### 3.2菜单选择

#### 3.2.1主菜单

通过主菜单选择所用功能。

// 主界面

Menu::Menu() {

Init();

while (true) { // 初始化菜单显示

system("cls");

cout << "------------------------------------------" << endl;

cout << "-------------江苏大学校园公交系统-------------" << endl;

cout << "--1.管理员登陆------------------------------" << endl;

cout << "--2.学生查询--------------------------------" << endl;

cout << "--3.写出-----------------------------------" << endl;

cout << "--4.退出-----------------------------------" << endl;

cout << "------------------------------------------" << endl;

cout << "Please enter a serial number to select: ";

char num;

cin >> num;

switch (num) {

case '1':

LoginAdmin();

break;

case '2':

StudentMenu();

break;

case '3':

Write();

break;

case '4':

exit(0);

default:

cout << " Error! No such service yet!" << endl;

sleep(2);

Menu();

}

}

}

#### 3.2.2管理员菜单

管理员系统使用需要先登陆已经身份验证处理

// 管理员登陆

bool Menu::LoginAdmin() {

system("cls");

cout << "------------------------------------------" << endl;

cout << "----------------Adminlogin----------------" << endl;

cout << "AdminUsername: ";

cin >> this->UserName;

cout << "AdminPassword: ";

cin >> this->Password;

cout << "------------------------------------------" << endl;

cout << "confirm(y/n): ";

string flag;

cin >> flag;

if (flag == "n") {

system("cls");

return false;

}

cout << "Loading………………" << endl;

sleep(3);

if (VerifyIdentidy()) {

cout << "------------Login Success!-------------" << endl;

sleep(1);

AdminMenu();

}

return false;

}

// 身份验证

bool Menu::VerifyIdentidy() {

if (this->UserName == "root") {

if (this->Password == "123456")

return true;

else {

cout << "Password ERROR!!!" << endl;

sleep(1);

LoginAdmin();

return false;

}

} else {

cout << "Admin is not exist !!!" << endl;

sleep(1);

return false;

}

}

管理员菜单有多种功能，修改公交线路、查询路线、退出系统。通过选择进入不同功能区。

// 管理员界面

bool Menu::AdminMenu() {

system("cls");

cout << "------------------------------------------" << endl;

cout << "---------------Administrator--------------" << endl;

cout << "--1.初始化---------------------------------" << endl;

cout << "--2.维护公交-------------------------------" << endl;

cout << "--3.返回----------------------------------" << endl;

cout << "------------------------------------------" << endl;

cout << "Please enter a serial number to select: ";

char num;

cin >> num;

switch (num) {

case '1': {

Init();

cout << "-------------Success-----------------" << endl;

sleep(3);

AdminMenu();

break;

}

case '2':

MaintainMenu();

break;

case '3':

break;

default:

cout << " Error! No such service yet!" << endl;

sleep(1);

system("cls");

AdminMenu();

}

return false;

}

下面是修改公交线路的菜单，主要有修改原有路线、新增路线、删除原有路线、站点的增删改以及返回到上一层菜单这几个功能。

bool Menu::MaintainMenu() {

system("cls");

cout << "------------------------------------------" << endl;

cout << "---------------MaintainBus----------------" << endl;

cout << "--1.新增公交线路----------------------------" << endl;

cout << "--2.删除公交线路----------------------------" << endl;

cout << "--3.修改已有公交线路-------------------------" << endl;

cout << "--4.删除已有站点----------------------------" << endl;

cout << "--5.增加站点，添加到已有公交线路中-------------" << endl;

cout << "--6.back----------------------------------" << endl;

cout << "------------------------------------------" << endl;

cout << "Please enter a serial number to select: ";

char flag;

cin >> flag;

switch (flag) {

case ' ':

break;

case '1':

AddBusRoute();

break;

case '2':

DeleteRoute();

break;

case '3':

ChangeBusRoute();

break;

case '4':

DeleteBusStation();

break;

case '5':

AddBusStation();

break;

case '6':

AdminMenu();

break;

default:

cout << " Error! No such service yet!" << endl;

sleep(1);

system("cls");

MaintainMenu();

}

return true;

}

// 添加公交路线

bool Menu::AddBusRoute() {

system("cls");

PrintRoute();

PrintStation();

cout << "---------------Add Route------------------" << endl;

cout << "Please input the bus name: ";

string NewBusName;

cin >> NewBusName;

while (RoutExist(NewBusName)) {

cout << "ERROR！The line already exists!" << endl;

cout << "Please input the bus name again : ";

cin >> NewBusName;

}

int OldStationNum;

Bus NewBus;

NewBus.StationNum = 0;

cout << "Please input the StationNum,if you want to end input '21' : ";

cin >> OldStationNum;

while (OldStationNum != 21 && OldStationNum < existStation.size) {

NewBus.StationName.StationName[NewBus.StationNum++] = existStation.ExistStation[OldStationNum];

cout << "Please input the next StationNum,if you want to end input '21' : ";

int NewStationNum;

cin >> NewStationNum;

if (NewStationNum == 21 || NewStationNum > existStation.size) break;

cout << "Please input the weight: ";

cin >> BusGraph.arc[OldStationNum][NewStationNum];

OldStationNum = NewStationNum;

}

cout << "Please input the price of route: ";

cin >> NewBus.money;

cout << "Loading Add………………" << endl;

BusMessage.push\_back(make\_pair(NewBusName, NewBus));

sleep(3);

cout << "---------------Add Success--------------" << endl;

PrintRoute();

sleep(3);

MaintainMenu();

return true;

}

// 删除公交路线

bool Menu::DeleteRoute() {

system("cls");

PrintRoute();

cout << "----------Delete Route----------" << endl;

cout << "Please input the bus name where you want to delete: ";

string deltebusName;

cin >> deltebusName;

while (!RoutExist(deltebusName)) {

cout << "The route not exists please input again: ";

cin >> deltebusName;

}

vector<pair<string, Bus>> NewBusMessage;

for (auto item: BusMessage) {

if (item.first != deltebusName) {

NewBusMessage.push\_back(item);

} else {

for (int i = 0; i < item.second.StationNum - 1; ++i) {

int flag = 0;

int nextflag = 0;

for (int j = 0; j < existStation.size; ++j) {

if (existStation.ExistStation[j] == item.second.StationName.StationName[i]) flag = j;

if (existStation.ExistStation[j] == item.second.StationName.StationName[i + 1]) nextflag = j;

}

BusGraph.arc[flag][nextflag] = INFINITY;

}

}

}

BusMessage = NewBusMessage;

cout << "-------Delete Success---------" << endl;

PrintRoute();

sleep(3);

MaintainMenu();

return true;

}

下面是修改原有路线的菜单，主要是修改路线名、修改行车路线、修改返回上一层这几个功能。

// 修改公交路线

bool Menu::ChangeBusRoute() {

system("cls");

cout << "------------------------------------------" << endl;

PrintRoute();

cout << "-------------Change Route-----------------" << endl;

PrintStation();

cout << "Please input the bus name : ";

string OldBusName;

cin >> OldBusName;

while (!RoutExist(OldBusName)) {

cout << "ERROR！The line not exists!" << endl;

cout << "Please input the bus name again : ";

cin >> OldBusName;

}

cout << "The Bus " << OldBusName << " has station: ";

for (auto item: BusMessage) {

if (item.first == OldBusName) {

for (int i = 0; i < item.second.StationNum; ++i) {

cout << i << " " << item.second.StationName.StationName[i] << " ";

}

cout << endl;

break;

}

}

cout << "please input the station number where you want to change:";

int num;

cin >> num;

int newnum;

cout << " Please input new number :";

cin >> newnum;

for (auto item: BusMessage) {

if (item.first == OldBusName) {

int lastflag, nextflag, flag;

for (int i = 0; i < existStation.size; ++i) {

if (item.second.StationName.StationName[num] == existStation.ExistStation[i]) {

flag = i;

}

if (item.second.StationName.StationName[num - 1] == existStation.ExistStation[i]) {

lastflag = i;

}

if (item.second.StationName.StationName[num + 1] == existStation.ExistStation[i]) {

nextflag = i;

}

}

BusGraph.arc[lastflag][newnum] = BusGraph.arc[lastflag][flag];

BusGraph.arc[newnum][nextflag] = BusGraph.arc[flag][nextflag];

BusGraph.arc[lastflag][flag] = INFINITY;

BusGraph.arc[flag][nextflag] = INFINITY;

item.second.StationName.StationName[num] = existStation.ExistStation[newnum];

break;

}

}

cout << "Loading Change……" << endl;

sleep(2);

cout << "-------------Change Success-------------" << endl;

PrintRoute();

sleep(3);

MaintainMenu();

return true;

}

下面是路线修改菜单，有站点添加、站点删除、返回菜单几个功能。

// 删除站点

bool Menu::DeleteBusStation() {

system("cls");

PrintRoute();

PrintStation();

cout << "Please input the station you want to delete : ";

int stationNum;

cin >> stationNum;

cout << "please tell me if you want to delete the station in all route(y/n) :";

char y;

cin >> y;

if (y == 'n') {

cout << "Please input where route you want to delete : ";

string routeName;

cin >> routeName;

while (!RoutExist(routeName)) {

cout << "ERROR！The line not exists!" << endl;

cout << "Please input the bus name again : ";

cin >> routeName;

}

for (auto item: BusMessage) {

if (item.first == routeName) {

int lastflag, nextflag, flag;

for (int i = 0; i < existStation.size; ++i) {

if (item.second.StationName.StationName[i] == existStation.ExistStation[stationNum]) {

flag = i;

lastflag = i - 1;

nextflag = i + 1;

}

}

for (int i = flag; i < item.second.StationNum - 1; ++i) {

item.second.StationName.StationName[i] = item.second.StationName.StationName[i + 1];

}

BusGraph.arc[lastflag][nextflag] = BusGraph.arc[lastflag][flag] + BusGraph.arc[flag][nextflag];

BusGraph.arc[lastflag][flag] = INFINITY;

BusGraph.arc[flag][nextflag] = INFINITY;

BusGraph.numEdges -= 2;

break;

}

}

} else if (y == 'y') {

for (auto item: BusMessage) {

int lastflag, nextflag, flag;

for (int i = 0; i < existStation.size; ++i) {

if (item.second.StationName.StationName[i] == existStation.ExistStation[stationNum]) {

flag = i;

lastflag = i - 1;

nextflag = i + 1;

}

}

for (int i = flag; i < item.second.StationNum - 1; ++i) {

item.second.StationName.StationName[i] = item.second.StationName.StationName[i + 1];

}

BusGraph.arc[lastflag][nextflag] = BusGraph.arc[lastflag][flag] + BusGraph.arc[flag][nextflag];

BusGraph.arc[lastflag][flag] = INFINITY;

BusGraph.arc[flag][nextflag] = INFINITY;

BusGraph.numEdges -= 2;

}

for (int i = stationNum; i < existStation.size - 1; ++i) {

existStation.ExistStation[i] = existStation.ExistStation[i + 1];

}

existStation.size--;

for (int i = stationNum; i < BusGraph.numVertexes - 1; ++i) {

for (int j = 0; j < BusGraph.numVertexes; ++j) {

BusGraph.arc[i][j] = BusGraph.arc[i + 1][j];

BusGraph.arc[j][i] = BusGraph.arc[j][i + 1];

}

}

BusGraph.numVertexes--;

}

cout << "--------------Delete Success---------" << endl;

PrintRoute();

sleep(3);

MaintainMenu();

return true;

}

// 添加新站点

bool Menu::AddBusStation() {

system("cls");

PrintStation();

cout << "-------Add BusStation-----" << endl;

string StationName;

cout << "please input StationName: ";

cin >> StationName;

while (StationExist(StationName)) {

cout << "This Staton already exist" << endl;

cout << "please input again :";

cin >> StationName;

}

existStation.ExistStation[existStation.size++] = StationName;

cout << " Add Success" << endl;

PrintRoute();

sleep(3);

MaintainMenu();

return true;

}

#### 3.3.3学生菜单

bool Menu::StudentMenu() {

system("cls");

cout << "-----------------------------------" << endl;

cout << "-----------Student-----------------" << endl;

cout << "1. 公交线路查询" << endl;

cout << "2. 公交路线规划" << endl;

cout << "3. 最优路线规划" << endl;

cout << "4. 环线" << endl;

cout << "5. back" << endl;

cout << "Please enter a serial number to select: ";

char num;

cin >> num;

switch (num) {

case '1':

FindRouteMenu();

break;

case '2':

RoutePlanMenu();

break;

case '3':

FiveTreeRoute();

break;

case '4':

HuanRoute();

break;

case '5':

break;

default:

cout << "ERROR!!!" << endl;

sleep(3);

exit(0);

}

return true;

}

#### 3.3.4 查询系统

下面是查询公交路线的菜单，主要有查询所有公交路线、查询某个站点公交路线、换乘查询、五棵松旅客路线查询、步行查询、校内环线公交以及返回上一层菜单的功能。

##### 3.3.4.1 五棵松到各地方的最短路线

查询从五棵松出发到达校内其他站点的最短路径以及所需时间。

bool Menu::FiveTreeRoute() {

system("cls");

int startflag;

for (int i = 0; i < existStation.size; ++i) {

if (existStation.ExistStation[i] == "五棵松") {

startflag = i;

break;

}

}

int U[WalkGraph.numVertexes];

int S[WalkGraph.numVertexes];

int Path[WalkGraph.numVertexes];

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

Path[i] = 0;

S[i] = 0;

U[i] = WalkGraph.arc[startflag][i];

}

S[startflag] = 1;

U[startflag] = 0;

int min\_num;

int min;

Path[startflag] = -1;

for (int i = 0; i < WalkGraph.numVertexes - 1; ++i) {

min = INFINITY;

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

if (!S[j] && U[j] < min) {

min = U[j];

min\_num = j;

}

}

S[min\_num] = 1;

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

if (U[j] > min + WalkGraph.arc[min\_num][j]) {

Path[j] = min\_num;

U[j] = min + WalkGraph.arc[min\_num][j];

}

}

}

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

if (Path[i] == 0) Path[i] = startflag;

}

stack<int> q;

for (int i = 0; i < WalkGraph.numVertexes - 1; ++i) {

int j = i;

while (Path[j] != -1) {

q.push(j);

j = Path[j];

}

q.push(j);

cout << "到达" << existStation.ExistStation[i] << "需要经过" << endl;

while (!q.empty()) {

cout << existStation.ExistStation[q.top()] << " ";

q.pop();

}

cout << " 所需时间:" << U[i] << " min" << endl;

}

cout << "you can input 'back': ";

string back;

while (true) {

cin >> back;

if (back == "back") StudentMenu();

else {

cout << "you must input 'back': ";

}

}

return true;

}

##### 3.3.4.2 公交线路查询界面

可以查询所有公交线路或者查询通过某个站点的所有的公交线路

bool Menu::FindRouteMenu() {

system("cls");

cout << "-------------------------" << endl;

cout << "1.查询站点" << endl;

cout << "2.查询所有路线" << endl;

cout << "3.back" << endl;

cout << "Please enter a serial number to select: ";

char num;

cin >> num;

switch (num) {

case '1':

FindStation();

break;

case '2': {

system("cls");

PrintRoute();

cout << "If you want to back please input 'back': ";

string back;

cin >> back;

while (back != "back") {

cout << "Error! If you must input 'back': ";

cin >> back;

}

FindRouteMenu();

break;

}

case '3':

StudentMenu();

break;

default:

cout << "ERROR!!!" << endl;

sleep(3);

FindRouteMenu();

}

return true;

}

// 查询站点信息

bool Menu::FindStation() {

system("cls");

PrintStation();

cout << "-----------Find Staiton--------" << endl;

int num;

cout << "please input the num where you want to find: ";

cin >> num;

cout << " 经过" << existStation.ExistStation[num] << "的路线有: " << endl;

for (auto item: BusMessage) {

for (int i = 0; i < item.second.StationNum; ++i) {

if (item.second.StationName.StationName[i] == existStation.ExistStation[num]) {

cout << item.first << endl;

break;

}

}

}

cout << "If you want to back please input 'back' or 'again': ";

string back;

cin >> back;

while (back != "back" && back != "again") {

cout << "Error! If you must input 'back' or 'again': ";

cin >> back;

}

if (back == "again") FindStation();

else FindRouteMenu();

return true;

}

// 线路规划界面

bool Menu::RoutePlanMenu() {

system("cls");

PrintStation();

cout << "----------Route Plan----------" << endl;

int SetOutNum, GotoNum;

cout << "Please input the station num where you in :";

cin >> SetOutNum;

cout << "Please input the station num where you want to go :";

cin >> GotoNum;

// 步行 最短

WalkMost(SetOutNum, GotoNum);

// 时间最短

TimingMost(SetOutNum, GotoNum);

cout << "you can input 'back' or 'again': ";

string back;

while (true) {

cin >> back;

if (back == "back") FindRouteMenu();

else if (back == "again") RoutePlanMenu();

else {

cout << "you must input 'back' or 'again': ";

}

}

return true;

}

##### 3.3.4.3 时间最短规划

从某个站点到某个站点的最短路径 直达、步行、一次换乘。

bool Menu::TimingMost(int SetoutNum, int GotoNum) {

cout << "乘坐公交车:" << endl;

int flag;

for (auto item: BusMessage) {

int startflag = INFINITY, endflag = INFINITY;

for (int i = 0; i < item.second.StationNum; ++i) {

if (item.second.StationName.StationName[i] == existStation.ExistStation[SetoutNum]) startflag = i;

if (item.second.StationName.StationName[i] == existStation.ExistStation[GotoNum]) endflag = i;

}

if (startflag != INFINITY && endflag != INFINITY) {

flag = 1;

cout << item.first << endl;

for (int i = 0; i < item.second.StationNum; ++i) {

cout << item.second.StationName.StationName[i] << " ";

}

cout << "票价为:" << item.second.money;

cout << endl;

}

}

if (flag == 0) {

cout << "不存在直达" << endl;

Transfer(SetoutNum, GotoNum);

}

return true;

}

// 换乘

bool Menu::Transfer(int SetOutNum, int GotoNUm) {

int startflag = SetOutNum;

int endflag = GotoNUm;

int midend=INFINITY,midbegin=INFINITY;

int beginbus=INFINITY,endbus=INFINITY;

int begin=INFINITY,end=INFINITY;

for (int i = 0; i < BusMessage.size(); ++i) {

midend=INFINITY;midbegin=INFINITY;

beginbus=INFINITY;endbus=INFINITY;

begin=INFINITY;end=INFINITY;

for (int j = 0; j < BusMessage[i].second.StationNum; ++j) {

if (existStation.ExistStation[startflag] == BusMessage[i].second.StationName.StationName[j]){

for (int k = 0; k < BusMessage.size(); ++k) {

for (int l = 0; l < BusMessage[k].second.StationNum; ++l) {

if (existStation.ExistStation[endflag] == BusMessage[k].second.StationName.StationName[l]){

for (int m = j; m < BusMessage[i].second.StationNum; ++m) {

for (int n = 0; n < l; ++n) {

if (BusMessage[i].second.StationName.StationName[m] == BusMessage[k].second.StationName.StationName[n]){

beginbus = i;

endbus = k;

begin = j;

end = l;

midbegin = m;

midend = n;

break;

}

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

cout << BusMessage[beginbus].first << BusMessage[endbus].first << " "<<endl;

cout << midend << midbegin;

return true;

}

// 步行最短

bool Menu::WalkMost(int SetOutNum, int GotoNum) {

cout << "步行距离最短:" << endl;

int startflag = SetOutNum;

int endflag = GotoNum;

int U[WalkGraph.numVertexes];

int S[WalkGraph.numVertexes];

int Path[WalkGraph.numVertexes];

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

Path[i] = 0;

S[i] = 0;

U[i] = WalkGraph.arc[startflag][i];

}

S[startflag] = 1;

U[startflag] = 0;

int min\_num;

int min;

Path[startflag] = -1;

for (int i = 0; i < WalkGraph.numVertexes - 1; ++i) {

min = INFINITY;

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

if (!S[j] && U[j] < min) {

min = U[j];

min\_num = j;

}

}

S[min\_num] = 1;

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

if (U[j] > min + WalkGraph.arc[min\_num][j]) {

Path[j] = min\_num;

U[j] = min + WalkGraph.arc[min\_num][j];

}

}

}

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

if (Path[i] == 0) Path[i] = startflag;

}

stack<int> q;

int j = endflag;

while (Path[j] != -1) {

q.push(j);

j = Path[j];

}

q.push(j);

cout << "从" << existStation.ExistStation[startflag] << " ";

cout << "步行到达 " << existStation.ExistStation[endflag] << "需要经过" << endl;

while (!q.empty()) {

cout << existStation.ExistStation[q.top()] << " ";

q.pop();

}

cout << " 步行所需时间:" << U[endflag] << " min" << endl;

return true;

}

##### 3.3.4.4 判断环线

输出校内存在的环线公交 即在一趟线路内存在两个相同站点的路线。

bool Menu::HuanRoute() {

system("cls");

vector<pair<string,Bus>> newBusMessage;

for (int i = 0; i < BusMessage.size()-1; ++i) {

for (int j = 0; j < BusMessage[i].second.StationNum; ++j) {

for (int k = j+1; k < BusMessage[i].second.StationNum; ++k) {

if (BusMessage[i].second.StationName.StationName[j] == BusMessage[i].second.StationName.StationName[k]){

newBusMessage.push\_back(BusMessage[i]);

}

}

}

}

cout << "环线有:" << endl;

for (auto item:newBusMessage) {

cout << item.first << ": ";

for (int i = 0; i < item.second.StationNum; ++i) {

cout << item.second.StationName.StationName[i] << " ";

}

cout << endl;

}

cout << "If you want to back please input 'back': ";

string back;

cin >> back;

while (back != "back") {

cout << "Error! If you must input 'back': ";

cin >> back;

}

StudentMenu();

return true;

}

## 四、运行结果及数据测试

运行截图如下

在主界面选择以学生查询还是管理员管理操作系统。

图片包含 图形用户界面

描述已自动生成

图4.1 程序主界面

###### 学生操作

用户输入 ’2’ 进入学生查询界面

图形用户界面, 文本

描述已自动生成

图4.2 学生界面

输入 ’1’ 查询公交线路

图片包含 文本

描述已自动生成

图4.3 查询界面

文本

描述已自动生成

图4.4 所有线路信息

用户输入站点编号，查询通过该站点的所有公交线路。

文本

描述已自动生成

图4.5 根据站点查询

用户输入两个地点的编号，查询直达，步行，单换乘的最短路径。

文本

描述已自动生成

图4.6 最短路径规划

从五棵松到校内各地最短路径

文本

描述已自动生成

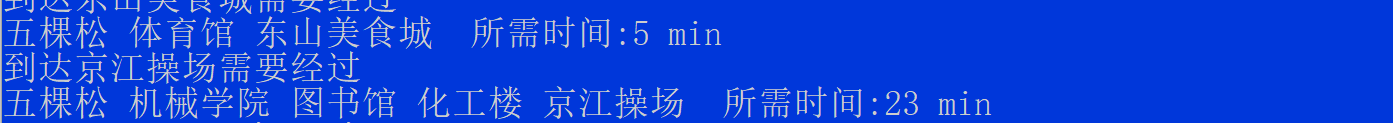


图4.7 最优路线规划



图4.8 校内环线公交

###### 管理员操作

图形用户界面, 文本

描述已自动生成

图4.9 管理员登录

日程表

描述已自动生成

图4.10 管理员界面

背景图案

中度可信度描述已自动生成

图4.11 路线维护界面

## 五、小结

本次课程设计，我个人感觉难度较大。一开始看到任务内容，我感觉并没有多难。直到课设周开始， 诸多课设搞得我手忙脚乱，不知所措。在刚开始我写好了一些基本功能，但是由于最初的构建过多的注重在功能的实现上，导致代码结构与数据结构严重冗余以致后续的功能实现造成严重困难，在最后对代码进行重构，时间严重不足导致功能不足全部实现。

一周四个课设的压力造成本次课设结果不是非常满意，从最初的设计构思到后续的功能实现都不是非常的符合逻辑，以及在最短路径算法的选择上也进行了纠结，但是最后依旧选择了Dijkstra(狄克斯特拉算法)查询最短路径，没有完全实现换乘功能是比较遗憾的，没有实现步行+公交换乘是让我感到非常沮丧的，我自认为我在程序设计上应该是有较好的能力去实现这几个功能的，但是没有在规定时间内完成这个项目，也充分说明了我的能力比较同级，同龄人来说是有非常巨大的进步空间的。

我相信在数据结构与算法这门课中的学习对我今后的帮助是非常巨大的，无论是Floyd-Warshall算法的原理：动态规划，还是Dijkstra算法的贪心算法的策略都使我的能力得到了不小的提升。

## 六、源码

### Menu.h

#ifndef COURSEDESIGN\_MENU\_H

#define COURSEDESIGN\_MENU\_H

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <typeinfo>

#define INFINITY 999999

using namespace std;

#define INFINITY 999999

class Menu {

private:

// 数据结构

typedef struct{

string StationName[10];

}Station;

typedef struct{

int StationNum; // 站台数目

Station StationName;

double money;

}Bus;

typedef struct{

string ExistStation[20]; // 站点名称

int size=0; // 已有站点的数量

}ExistStation;

// 图

typedef struct{

int numVertexes;

int numEdges;

int arc[20][20];

}MGraph;

ExistStation existStation;

MGraph WalkGraph,BusGraph;

vector<pair<string,Bus>> BusMessage; // 公交车信息

string UserName; // 用户名

string Password;// 密码

/\*\*/

/\* 方法\*/

vector<string> split(const string &s, const string &seperator);

bool Init();

bool LoginAdmin();

bool VerifyIdentidy();

bool AdminMenu();

bool MaintainMenu();

bool AddBusRoute();

bool PrintRoute();

bool PrintStation();

bool RoutExist(string BusName);

bool StationExist(string StationName);

bool DeleteRoute();

bool ChangeBusRoute();

bool DeleteBusStation();

bool AddBusStation();

bool StudentMenu();

bool FindRouteMenu();

bool FindStation();

bool RoutePlanMenu();

bool WalkMost(int SetoutNum,int GotoNum);

bool TimingMost(int SetoutNum,int GotoNum);

bool FiveTreeRoute();

bool Transfer(int SetOutNum,int GotoNUm);

bool Write();

bool HuanRoute();

public:

Menu();

};

#endif //COURSEDESIGN\_MENU\_H

### Menu.cpp

#include "Menu.h"

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <typeinfo>

#include <unistd.h>

#include <stack>

#define INFINITY 999999

using namespace std;

// 读文件时数据处理

vector<string> Menu::split(const string &s, const string &seperator) {

vector<string> result;

int i = 0;

while (i != s.size()) {

//找到字符串中首个不等于分隔符的字母；

int flag = 0;

while (i != s.size() && flag == 0) {

flag = 1;

for (int x = 0; x < seperator.size(); ++x) {

if (s[i] == seperator[x]) {

++i;

flag = 0;

break;

}

}

}

flag = 0;

int j = i;

while (j != s.size() && flag == 0) {

for (int x = 0; x < seperator.size(); ++x) {

if (s[j] == seperator[x]) {

flag = 1;

break;

}

}

if (flag == 0) ++j;

}

if (i != j) {

result.push\_back(string(s.substr(i, j - i)));

i = j;

}

}

return result;

}

// 读文件初始化

bool Menu::Init() {

ifstream ifs;

vector<string> tokens;

tokens.clear();

existStation.size = 0;

BusGraph.numEdges = 0;

BusGraph.numVertexes = 0;

WalkGraph.numVertexes = 0;

WalkGraph.numEdges = 0;

ifs.open("E:\\ujs\_meng\\Shuju\_struct\_c++\\CourseDesign\\WalkRoutes.txt", ios::in);

string s;

// 添加所有站点

getline(ifs, s);

do {

tokens.push\_back(s);

getline(ifs, s);

} while (s != "//");

tokens = split(tokens[0], ",");

for (auto item: tokens) {

vector<string> test = split(item, " ");

existStation.ExistStation[existStation.size++] = test[1];

}

// 输出所有站点

// for (int i = 0; i < existStation.size; ++i) {

// cout << existStation.ExistStation[i] << " ";

// }

cout << endl;

// 添加公交线路信息

getline(ifs, s);

do {

tokens.clear();

tokens.push\_back(s);

getline(ifs, s);

vector<string> test = split(tokens[0], ":");

string buf = test[0];

Bus buff;

buff.money = stof(test[2]);

buff.StationNum = test[1].size() / 2 + 1;

int j = 0;

vector<string> in = split(test[1], ",");

int i = 0;

for (auto item: in) {

buff.StationName.StationName[i++] = existStation.ExistStation[stoi(item)];

}

BusMessage.push\_back(make\_pair(buf, buff));

} while (s != "//");

// 输出公交线路信息

// for (auto item:BusMessage) {

// cout << item.first << " ";

// for (int i = 0; i < item.second.StationNum; ++i) {

// cout << item.second.StationName.StationName[i] << " ";

// }

// cout << " 票价为: " << item.second.money << "元";

// cout << endl;

// }

// 添加公交邻接矩阵

BusGraph.numVertexes = existStation.size;

for (int i = 0; i < BusGraph.numVertexes; ++i) {

for (int j = 0; j < BusGraph.numVertexes; ++j) {

BusGraph.arc[i][j] = BusGraph.arc[j][i] = INFINITY;

if (i == j) {

BusGraph.arc[i][j] = BusGraph.arc[j][i] = 0;

}

}

}

getline(ifs, s);

do {

tokens.clear();

tokens = split(s, ",");

BusGraph.arc[stoi(tokens[1])][stoi(tokens[0])] = BusGraph.arc[stoi(tokens[0])][stoi(tokens[1])] = stoi(

tokens[2]);

BusGraph.numEdges++;

getline(ifs, s);

} while (s != "//");

// 输出公交邻接矩阵

// cout << "输出公交邻接矩阵" << endl;

// for (int i = 0; i < BusGraph.numVertexes; ++i) {

// for (int j = 0; j < BusGraph.numVertexes; ++j) {

// cout << BusGraph.arc[i][j] << " ";

// }

// cout << endl;

// }

// 添加步行邻接矩阵

WalkGraph.numVertexes = existStation.size;

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

WalkGraph.arc[i][j] = WalkGraph.arc[j][i] = INFINITY;

if (i == j) {

WalkGraph.arc[i][j] = WalkGraph.arc[j][i] = 0;

}

}

}

getline(ifs, s);

do {

tokens.clear();

tokens = split(s, ",");

WalkGraph.arc[stoi(tokens[0])][stoi(tokens[1])] = WalkGraph.arc[stoi(tokens[1])][stoi(tokens[0])] = stoi(

tokens[2]);

WalkGraph.numEdges++;

} while (getline(ifs, s));

// 输出步行邻接矩阵

// cout << "输出步行邻接矩阵" << endl;

// for (int i = 0; i < WalkGraph.numVertexes; ++i) {

// for (int j = 0; j < WalkGraph.numVertexes; ++j) {

// cout << WalkGraph.arc[i][j] << " ";

// }

// cout << endl;

// }

ifs.close();

return true;

}

// 主界面

Menu::Menu() {

Init();

while (true) { // 初始化菜单显示

system("cls");

cout << "------------------------------------------" << endl;

cout << "-------------江苏大学校园公交系统-------------" << endl;

cout << "--1.管理员登陆------------------------------" << endl;

cout << "--2.学生查询--------------------------------" << endl;

cout << "--3.写出-----------------------------------" << endl;

cout << "--4.退出-----------------------------------" << endl;

cout << "------------------------------------------" << endl;

cout << "Please enter a serial number to select: ";

char num;

cin >> num;

switch (num) {

case '1':

LoginAdmin();

break;

case '2':

StudentMenu();

break;

case '3':

Write();

break;

case '4':

exit(0);

default:

cout << " Error! No such service yet!" << endl;

sleep(2);

Menu();

}

}

}

// 学生界面

bool Menu::StudentMenu() {

system("cls");

cout << "-----------------------------------" << endl;

cout << "-----------Student-----------------" << endl;

cout << "1. 公交线路查询" << endl;

cout << "2. 公交路线规划" << endl;

cout << "3. 最优路线规划" << endl;

cout << "4. 环线" << endl;

cout << "5. back" << endl;

cout << "Please enter a serial number to select: ";

char num;

cin >> num;

switch (num) {

case '1':

FindRouteMenu();

break;

case '2':

RoutePlanMenu();

break;

case '3':

FiveTreeRoute();

break;

case '4':

HuanRoute();

break;

case '5':

break;

default:

cout << "ERROR!!!" << endl;

sleep(3);

exit(0);

}

return true;

}

// 五棵松到各地方的最短路线(不完善)

bool Menu::FiveTreeRoute() {

system("cls");

int startflag;

for (int i = 0; i < existStation.size; ++i) {

if (existStation.ExistStation[i] == "五棵松") {

startflag = i;

break;

}

}

int U[WalkGraph.numVertexes];

int S[WalkGraph.numVertexes];

int Path[WalkGraph.numVertexes];

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

Path[i] = 0;

S[i] = 0;

U[i] = WalkGraph.arc[startflag][i];

}

S[startflag] = 1;

U[startflag] = 0;

int min\_num;

int min;

Path[startflag] = -1;

for (int i = 0; i < WalkGraph.numVertexes - 1; ++i) {

min = INFINITY;

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

if (!S[j] && U[j] < min) {

min = U[j];

min\_num = j;

}

}

S[min\_num] = 1;

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

if (U[j] > min + WalkGraph.arc[min\_num][j]) {

Path[j] = min\_num;

U[j] = min + WalkGraph.arc[min\_num][j];

}

}

}

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

if (Path[i] == 0) Path[i] = startflag;

}

stack<int> q;

for (int i = 0; i < WalkGraph.numVertexes - 1; ++i) {

int j = i;

while (Path[j] != -1) {

q.push(j);

j = Path[j];

}

q.push(j);

cout << "到达" << existStation.ExistStation[i] << "需要经过" << endl;

while (!q.empty()) {

cout << existStation.ExistStation[q.top()] << " ";

q.pop();

}

cout << " 所需时间:" << U[i] << " min" << endl;

}

cout << "you can input 'back': ";

string back;

while (true) {

cin >> back;

if (back == "back") StudentMenu();

else {

cout << "you must input 'back': ";

}

}

return true;

}

// 公交线路查询界面

bool Menu::FindRouteMenu() {

system("cls");

cout << "-------------------------" << endl;

cout << "1.查询站点" << endl;

cout << "2.查询所有路线" << endl;

cout << "3.back" << endl;

cout << "Please enter a serial number to select: ";

char num;

cin >> num;

switch (num) {

case '1':

FindStation();

break;

case '2': {

system("cls");

PrintRoute();

cout << "If you want to back please input 'back': ";

string back;

cin >> back;

while (back != "back") {

cout << "Error! If you must input 'back': ";

cin >> back;

}

FindRouteMenu();

break;

}

case '3':

StudentMenu();

break;

default:

cout << "ERROR!!!" << endl;

sleep(3);

FindRouteMenu();

}

return true;

}

// 查询站点信息

bool Menu::FindStation() {

system("cls");

PrintStation();

cout << "-----------Find Staiton--------" << endl;

int num;

cout << "please input the num where you want to find: ";

cin >> num;

cout << " 经过" << existStation.ExistStation[num] << "的路线有: " << endl;

for (auto item: BusMessage) {

for (int i = 0; i < item.second.StationNum; ++i) {

if (item.second.StationName.StationName[i] == existStation.ExistStation[num]) {

cout << item.first << endl;

break;

}

}

}

cout << "If you want to back please input 'back' or 'again': ";

string back;

cin >> back;

while (back != "back" && back != "again") {

cout << "Error! If you must input 'back' or 'again': ";

cin >> back;

}

if (back == "again") FindStation();

else FindRouteMenu();

return true;

}

// 线路规划界面(未完成)

bool Menu::RoutePlanMenu() {

system("cls");

PrintStation();

cout << "----------Route Plan----------" << endl;

int SetOutNum, GotoNum;

cout << "Please input the station num where you in :";

cin >> SetOutNum;

cout << "Please input the station num where you want to go :";

cin >> GotoNum;

// 步行 最短

WalkMost(SetOutNum, GotoNum);

// 时间最短

TimingMost(SetOutNum, GotoNum);

cout << "you can input 'back' or 'again': ";

string back;

while (true) {

cin >> back;

if (back == "back") FindRouteMenu();

else if (back == "again") RoutePlanMenu();

else {

cout << "you must input 'back' or 'again': ";

}

}

return true;

}

// 时间最短

bool Menu::TimingMost(int SetoutNum, int GotoNum) {

cout << "乘坐公交车:" << endl;

int flag;

for (auto item: BusMessage) {

int startflag = INFINITY, endflag = INFINITY;

for (int i = 0; i < item.second.StationNum; ++i) {

if (item.second.StationName.StationName[i] == existStation.ExistStation[SetoutNum]) startflag = i;

if (item.second.StationName.StationName[i] == existStation.ExistStation[GotoNum]) endflag = i;

}

if (startflag != INFINITY && endflag != INFINITY) {

flag = 1;

cout << item.first << endl;

for (int i = 0; i < item.second.StationNum; ++i) {

cout << item.second.StationName.StationName[i] << " ";

}

cout << "票价为:" << item.second.money;

cout << endl;

}

}

if (flag == 0) {

cout << "不存在直达" << endl;

Transfer(SetoutNum, GotoNum);

}

return true;

}

// 换乘(未完成(待修改))

bool Menu::Transfer(int SetOutNum, int GotoNUm) {

int startflag = SetOutNum;

int endflag = GotoNUm;

int midend=INFINITY,midbegin=INFINITY;

int beginbus=INFINITY,endbus=INFINITY;

int begin=INFINITY,end=INFINITY;

for (int i = 0; i < BusMessage.size(); ++i) {

midend=INFINITY;midbegin=INFINITY;

beginbus=INFINITY;endbus=INFINITY;

begin=INFINITY;end=INFINITY;

for (int j = 0; j < BusMessage[i].second.StationNum; ++j) {

if (existStation.ExistStation[startflag] == BusMessage[i].second.StationName.StationName[j]){

for (int k = 0; k < BusMessage.size(); ++k) {

for (int l = 0; l < BusMessage[k].second.StationNum; ++l) {

if (existStation.ExistStation[endflag] == BusMessage[k].second.StationName.StationName[l]){

for (int m = j; m < BusMessage[i].second.StationNum; ++m) {

for (int n = 0; n < l; ++n) {

if (BusMessage[i].second.StationName.StationName[m] == BusMessage[k].second.StationName.StationName[n]){

beginbus = i;

endbus = k;

begin = j;

end = l;

midbegin = m;

midend = n;

break;

}

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

if (midbegin != INFINITY && midend != INFINITY) break;

}

cout << BusMessage[beginbus].first << BusMessage[endbus].first << " "<<endl;

cout << midend << midbegin;

return true;

}

// 步行最短

bool Menu::WalkMost(int SetOutNum, int GotoNum) {

cout << "步行距离最短:" << endl;

int startflag = SetOutNum;

int endflag = GotoNum;

int U[WalkGraph.numVertexes];

int S[WalkGraph.numVertexes];

int Path[WalkGraph.numVertexes];

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

Path[i] = 0;

S[i] = 0;

U[i] = WalkGraph.arc[startflag][i];

}

S[startflag] = 1;

U[startflag] = 0;

int min\_num;

int min;

Path[startflag] = -1;

for (int i = 0; i < WalkGraph.numVertexes - 1; ++i) {

min = INFINITY;

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

if (!S[j] && U[j] < min) {

min = U[j];

min\_num = j;

}

}

S[min\_num] = 1;

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

if (U[j] > min + WalkGraph.arc[min\_num][j]) {

Path[j] = min\_num;

U[j] = min + WalkGraph.arc[min\_num][j];

}

}

}

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

if (Path[i] == 0) Path[i] = startflag;

}

stack<int> q;

int j = endflag;

while (Path[j] != -1) {

q.push(j);

j = Path[j];

}

q.push(j);

cout << "从" << existStation.ExistStation[startflag] << " ";

cout << "步行到达 " << existStation.ExistStation[endflag] << "需要经过" << endl;

while (!q.empty()) {

cout << existStation.ExistStation[q.top()] << " ";

q.pop();

}

cout << " 步行所需时间:" << U[endflag] << " min" << endl;

return true;

}

// 判断环线

bool Menu::HuanRoute() {

system("cls");

vector<pair<string,Bus>> newBusMessage;

for (int i = 0; i < BusMessage.size()-1; ++i) {

for (int j = 0; j < BusMessage[i].second.StationNum; ++j) {

for (int k = j+1; k < BusMessage[i].second.StationNum; ++k) {

if (BusMessage[i].second.StationName.StationName[j] == BusMessage[i].second.StationName.StationName[k]){

newBusMessage.push\_back(BusMessage[i]);

}

}

}

}

cout << "环线有:" << endl;

for (auto item:newBusMessage) {

cout << item.first << ": ";

for (int i = 0; i < item.second.StationNum; ++i) {

cout << item.second.StationName.StationName[i] << " ";

}

cout << endl;

}

cout << "If you want to back please input 'back': ";

string back;

cin >> back;

while (back != "back") {

cout << "Error! If you must input 'back': ";

cin >> back;

}

StudentMenu();

return true;

}

// 管理员登陆

bool Menu::LoginAdmin() {

system("cls");

cout << "------------------------------------------" << endl;

cout << "----------------Adminlogin----------------" << endl;

cout << "AdminUsername: ";

cin >> this->UserName;

cout << "AdminPassword: ";

cin >> this->Password;

cout << "------------------------------------------" << endl;

cout << "confirm(y/n): ";

string flag;

cin >> flag;

if (flag == "n") {

system("cls");

return false;

}

cout << "Loading………………" << endl;

sleep(3);

if (VerifyIdentidy()) {

cout << "------------Login Success!-------------" << endl;

sleep(1);

AdminMenu();

}

return false;

}

// 身份验证

bool Menu::VerifyIdentidy() {

if (this->UserName == "root") {

if (this->Password == "123456")

return true;

else {

cout << "Password ERROR!!!" << endl;

sleep(1);

LoginAdmin();

return false;

}

} else {

cout << "Admin is not exist !!!" << endl;

sleep(1);

return false;

}

}

// 管理员界面

bool Menu::AdminMenu() {

system("cls");

cout << "------------------------------------------" << endl;

cout << "---------------Administrator--------------" << endl;

cout << "--1.初始化---------------------------------" << endl;

cout << "--2.维护公交-------------------------------" << endl;

cout << "--3.返回----------------------------------" << endl;

cout << "------------------------------------------" << endl;

cout << "Please enter a serial number to select: ";

char num;

cin >> num;

switch (num) {

case '1': {

Init();

cout << "-------------Success-----------------" << endl;

sleep(3);

AdminMenu();

break;

}

case '2':

MaintainMenu();

break;

case '3':

break;

default:

cout << " Error! No such service yet!" << endl;

sleep(1);

system("cls");

AdminMenu();

}

return false;

}

// 维护公交

bool Menu::MaintainMenu() {

system("cls");

cout << "------------------------------------------" << endl;

cout << "---------------MaintainBus----------------" << endl;

cout << "--1.新增公交线路----------------------------" << endl;

cout << "--2.删除公交线路----------------------------" << endl;

cout << "--3.修改已有公交线路-------------------------" << endl;

cout << "--4.删除已有站点----------------------------" << endl;

cout << "--5.增加站点，添加到已有公交线路中-------------" << endl;

cout << "--6.back----------------------------------" << endl;

cout << "------------------------------------------" << endl;

cout << "Please enter a serial number to select: ";

char flag;

cin >> flag;

switch (flag) {

case ' ':

break;

case '1':

AddBusRoute();

break;

case '2':

DeleteRoute();

break;

case '3':

ChangeBusRoute();

break;

case '4':

DeleteBusStation();

break;

case '5':

AddBusStation();

break;

case '6':

AdminMenu();

break;

default:

cout << " Error! No such service yet!" << endl;

sleep(1);

system("cls");

MaintainMenu();

}

return true;

}

// 添加公交路线

bool Menu::AddBusRoute() {

system("cls");

PrintRoute();

PrintStation();

cout << "---------------Add Route------------------" << endl;

cout << "Please input the bus name: ";

string NewBusName;

cin >> NewBusName;

while (RoutExist(NewBusName)) {

cout << "ERROR！The line already exists!" << endl;

cout << "Please input the bus name again : ";

cin >> NewBusName;

}

int OldStationNum;

Bus NewBus;

NewBus.StationNum = 0;

cout << "Please input the StationNum,if you want to end input '21' : ";

cin >> OldStationNum;

while (OldStationNum != 21 && OldStationNum < existStation.size) {

NewBus.StationName.StationName[NewBus.StationNum++] = existStation.ExistStation[OldStationNum];

cout << "Please input the next StationNum,if you want to end input '21' : ";

int NewStationNum;

cin >> NewStationNum;

if (NewStationNum == 21 || NewStationNum > existStation.size) break;

cout << "Please input the weight: ";

cin >> BusGraph.arc[OldStationNum][NewStationNum];

OldStationNum = NewStationNum;

}

cout << "Please input the price of route: ";

cin >> NewBus.money;

cout << "Loading Add………………" << endl;

BusMessage.push\_back(make\_pair(NewBusName, NewBus));

sleep(3);

cout << "---------------Add Success--------------" << endl;

PrintRoute();

sleep(3);

MaintainMenu();

return true;

}

// 删除公交路线

bool Menu::DeleteRoute() {

system("cls");

PrintRoute();

cout << "----------Delete Route----------" << endl;

cout << "Please input the bus name where you want to delete: ";

string deltebusName;

cin >> deltebusName;

while (!RoutExist(deltebusName)) {

cout << "The route not exists please input again: ";

cin >> deltebusName;

}

vector<pair<string, Bus>> NewBusMessage;

for (auto item: BusMessage) {

if (item.first != deltebusName) {

NewBusMessage.push\_back(item);

} else {

for (int i = 0; i < item.second.StationNum - 1; ++i) {

int flag = 0;

int nextflag = 0;

for (int j = 0; j < existStation.size; ++j) {

if (existStation.ExistStation[j] == item.second.StationName.StationName[i]) flag = j;

if (existStation.ExistStation[j] == item.second.StationName.StationName[i + 1]) nextflag = j;

}

BusGraph.arc[flag][nextflag] = INFINITY;

}

}

}

BusMessage = NewBusMessage;

cout << "-------Delete Success---------" << endl;

PrintRoute();

sleep(3);

MaintainMenu();

return true;

}

// 修改公交路线

bool Menu::ChangeBusRoute() {

system("cls");

cout << "------------------------------------------" << endl;

PrintRoute();

cout << "-------------Change Route-----------------" << endl;

PrintStation();

cout << "Please input the bus name : ";

string OldBusName;

cin >> OldBusName;

while (!RoutExist(OldBusName)) {

cout << "ERROR！The line not exists!" << endl;

cout << "Please input the bus name again : ";

cin >> OldBusName;

}

cout << "The Bus " << OldBusName << " has station: ";

for (auto item: BusMessage) {

if (item.first == OldBusName) {

for (int i = 0; i < item.second.StationNum; ++i) {

cout << i << " " << item.second.StationName.StationName[i] << " ";

}

cout << endl;

break;

}

}

cout << "please input the station number where you want to change:";

int num;

cin >> num;

int newnum;

cout << " Please input new number :";

cin >> newnum;

for (auto item: BusMessage) {

if (item.first == OldBusName) {

int lastflag, nextflag, flag;

for (int i = 0; i < existStation.size; ++i) {

if (item.second.StationName.StationName[num] == existStation.ExistStation[i]) {

flag = i;

}

if (item.second.StationName.StationName[num - 1] == existStation.ExistStation[i]) {

lastflag = i;

}

if (item.second.StationName.StationName[num + 1] == existStation.ExistStation[i]) {

nextflag = i;

}

}

BusGraph.arc[lastflag][newnum] = BusGraph.arc[lastflag][flag];

BusGraph.arc[newnum][nextflag] = BusGraph.arc[flag][nextflag];

BusGraph.arc[lastflag][flag] = INFINITY;

BusGraph.arc[flag][nextflag] = INFINITY;

item.second.StationName.StationName[num] = existStation.ExistStation[newnum];

break;

}

}

cout << "Loading Change……" << endl;

sleep(2);

cout << "-------------Change Success-------------" << endl;

PrintRoute();

sleep(3);

MaintainMenu();

return true;

}

// 删除站点

bool Menu::DeleteBusStation() {

system("cls");

PrintRoute();

PrintStation();

cout << "Please input the station you want to delete : ";

int stationNum;

cin >> stationNum;

cout << "please tell me if you want to delete the station in all route(y/n) :";

char y;

cin >> y;

if (y == 'n') {

cout << "Please input where route you want to delete : ";

string routeName;

cin >> routeName;

while (!RoutExist(routeName)) {

cout << "ERROR！The line not exists!" << endl;

cout << "Please input the bus name again : ";

cin >> routeName;

}

for (auto item: BusMessage) {

if (item.first == routeName) {

int lastflag, nextflag, flag;

for (int i = 0; i < existStation.size; ++i) {

if (item.second.StationName.StationName[i] == existStation.ExistStation[stationNum]) {

flag = i;

lastflag = i - 1;

nextflag = i + 1;

}

}

for (int i = flag; i < item.second.StationNum - 1; ++i) {

item.second.StationName.StationName[i] = item.second.StationName.StationName[i + 1];

}

BusGraph.arc[lastflag][nextflag] = BusGraph.arc[lastflag][flag] + BusGraph.arc[flag][nextflag];

BusGraph.arc[lastflag][flag] = INFINITY;

BusGraph.arc[flag][nextflag] = INFINITY;

BusGraph.numEdges -= 2;

break;

}

}

} else if (y == 'y') {

for (auto item: BusMessage) {

int lastflag, nextflag, flag;

for (int i = 0; i < existStation.size; ++i) {

if (item.second.StationName.StationName[i] == existStation.ExistStation[stationNum]) {

flag = i;

lastflag = i - 1;

nextflag = i + 1;

}

}

for (int i = flag; i < item.second.StationNum - 1; ++i) {

item.second.StationName.StationName[i] = item.second.StationName.StationName[i + 1];

}

BusGraph.arc[lastflag][nextflag] = BusGraph.arc[lastflag][flag] + BusGraph.arc[flag][nextflag];

BusGraph.arc[lastflag][flag] = INFINITY;

BusGraph.arc[flag][nextflag] = INFINITY;

BusGraph.numEdges -= 2;

}

for (int i = stationNum; i < existStation.size - 1; ++i) {

existStation.ExistStation[i] = existStation.ExistStation[i + 1];

}

existStation.size--;

for (int i = stationNum; i < BusGraph.numVertexes - 1; ++i) {

for (int j = 0; j < BusGraph.numVertexes; ++j) {

BusGraph.arc[i][j] = BusGraph.arc[i + 1][j];

BusGraph.arc[j][i] = BusGraph.arc[j][i + 1];

}

}

BusGraph.numVertexes--;

}

cout << "--------------Delete Success---------" << endl;

PrintRoute();

sleep(3);

MaintainMenu();

return true;

}

// 添加新站点

bool Menu::AddBusStation() {

system("cls");

PrintStation();

cout << "-------Add BusStation-----" << endl;

string StationName;

cout << "please input StationName: ";

cin >> StationName;

while (StationExist(StationName)) {

cout << "This Staton already exist" << endl;

cout << "please input again :";

cin >> StationName;

}

existStation.ExistStation[existStation.size++] = StationName;

cout << " Add Success" << endl;

PrintRoute();

sleep(3);

MaintainMenu();

return true;

}

// 输出所有公交线路信息

bool Menu::PrintRoute() {

cout << "所有公交线路包括: " << endl;

for (auto item: BusMessage) {

cout << item.first << " ";

for (int i = 0; i < item.second.StationNum; ++i) {

cout << item.second.StationName.StationName[i] << " ";

}

cout << " 票价为: " << item.second.money << "元";

cout << endl;

}

return true;

}

// 输出所有公交站点信息

bool Menu::PrintStation() {

cout << "所有站点信息: " << endl;

for (int i = 0; i < existStation.size; ++i) {

cout << i << " " << existStation.ExistStation[i] << " ";

}

cout << endl;

return true;

}

// 查找线路是否存在

bool Menu::RoutExist(string BusName) {

for (auto item = BusMessage.begin(); item != BusMessage.end(); ++item) {

if (item->first == BusName) return true;

}

return false;

}

// 站点是否存在

bool Menu::StationExist(string StationName) {

for (auto item: existStation.ExistStation) {

if (item == StationName) {

return true;

}

}

return false;

}

// 写文件

bool Menu::Write() {

ofstream ofs;

ofs.open("E:\\ujs\_meng\\Shuju\_struct\_c++\\CourseDesign\\NewRoutes.txt",ios::out);

for (int i = 0; i < existStation.size-1; ++i) {

ofs << i << " " << existStation.ExistStation[i] << ",";

}

ofs << existStation.size << existStation.ExistStation[existStation.size] << endl;

ofs << "//" << endl;

for (int i = 0; i < BusGraph.numVertexes; ++i) {

for (int j = 0; j < BusGraph.numVertexes; ++j) {

if (BusGraph.arc[i][j] != INFINITY && i != j){

ofs << i << "," << j << "," << BusGraph.arc[i][j] << endl;

}

}

}

ofs << "//" << endl;

for (int i = 0; i < WalkGraph.numVertexes; ++i) {

for (int j = 0; j < WalkGraph.numVertexes; ++j) {

if (WalkGraph.arc[i][j] != INFINITY && i != j){

ofs << i << "," << j << "," << WalkGraph.arc[i][j] << endl;

}

}

}

ofs.close();

return true;

}

### Main.cpp

#include <iostream>

#include "Menu.cpp"

using namespace std;

int main() {

Menu s;

return 0;

}

### WalkRoutes.txt

0 六食堂, 1 新一区, 2 图书馆, 3 三岔口, 4 讲堂群, 5 五棵松, 6 京江楼, 7 机械学院, 8 化工楼, 9 体育馆, 10 食品学院, 11 小礼堂, 12 东山美食城, 13 京江操场, 14 校医院

//

一号线:0,2,5,4,3,1,0:1.0

二号线:0,1,3,4,5,2,8,6,0:1.5

三号线:0,1,13,4,5,9,12:1.5

四号线:0,1,3,10,11:1.0

五号线:0,13,2,7,5,12,14:1.5

//

0,1,3

0,6,2

0,13,3

1,3,4

2,7,1

2,8,1

2,13,6

3,4,2

3,10,2

4,5,2

5,7,5

5,9,2

6,8,7

9,12,1

10,11,3

12,14,2

//

0,1,10

0,6,5

1,3,7

3,4,2

4,5,2

4,7,6

5,9,3

9,12,2

12,14,8

12,10,6

10,11,2

3,10,3

6,13,2

13,8,10

8,2,2

2,7,3

7,5,8