**计算机科学与技术学院课程设计成绩单**

课程名称：数据结构课程设计

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 何浩 | 性别 | 男 | 学号 | | 201913136019 | | 班级 | 网络1901班 |
| 电话 | 15807196050 | | 综合成绩 | |  | | 成绩等级 | |  |
| 程序运行情况  （占总成绩20%） | |  | | | | | | | |
| 程序功能完善程度  （占总成绩20%） | |  | | | | | | | |
| 对问题的答辩情况  （占总成绩40%） | |  | | | | | | | |
| 学生的工作态度与独立工作能力  （占总成绩10%） | |  | | | | | | | |
| 设计报告的规范性  （占总成绩10%） | |  | | | | | | | |

A：90~100分 A-：85~89分 B+：82~84分 B：78~81分 B-：75~77分

C+：72~74分 C：68~71分 C-：64~67分 D：60~63分 F：<60分

武汉科技大学计算机科学与技术学院制表

目录

[[题目] 公交线路图的构建和查询 2](#_Toc65938113)

[0.流程图和功能介绍 4](#_Toc65938114)

[1.设计特点 6](#_Toc65938115)

[2.程序整体结构及类的函数 7](#_Toc65938116)

[3.数据格式 9](#_Toc65938117)

[4.数据结构（读文件创建图） 10](#_Toc65938118)

[5.关键算法描述 13](#_Toc65938119)

[6.查询公交线路和站点信息 20](#_Toc65938120)

[7.查询两站点之间的路线，找到至多换乘1次的路线，并输出结果。 22](#_Toc65938121)

[8.修改公交线路和站点信息，保存文件 24](#_Toc65938122)

[9.设计总结 27](#_Toc65938123)

## [题目] 公交线路图的构建和查询

参考《数据结构实践教程》P56，所有功能如教程所示，并附加文件存储功能，即站点消息、公交线路信息等数据不得写入程序中，要求保存在文件中。

具体说明如下：

1、用文件一，例如文本文件，文件名stations.txt，保存站点信息，**格式可以自行设计**，例如：编号 站名

1 黄家湖西路武科大

2 黄家湖西路岐黄南路

3 黄家湖西路中医药大学

……

20 和平大道三角路

……

2、用文件二，例如文本文件，文件名buses.txt，保存公交信息，**格式可以自行设计**，例如：编号 线路名 起点站编号 终点站编号

1 539上行 1 20

2 539下行 20 1

……

3、用文件三，例如文本文件，文件名routes.txt，保存线路信息，**格式可以自行设计**，例如：线路编号 站点编号 站点编号 距离

1 1 2 650

1 2 3 400

……

意思是：

539上行 （从）黄家湖西路武科大 （到）黄家湖西路岐黄南路 650米

539上行 （从）黄家湖西路岐黄南路 （到）黄家湖西路中医药大学 400米

……

4、设计公交线路所需的存储结构，将文件中的数据读入内存。

5、提供用户操作的菜单和界面实现添加、删除、修改公交、站点、线路信息，注意数据间的关联性。

6、将修改后的信息保存回文件。

7、其他查询操作参见教程。

## 0.流程图和功能介绍

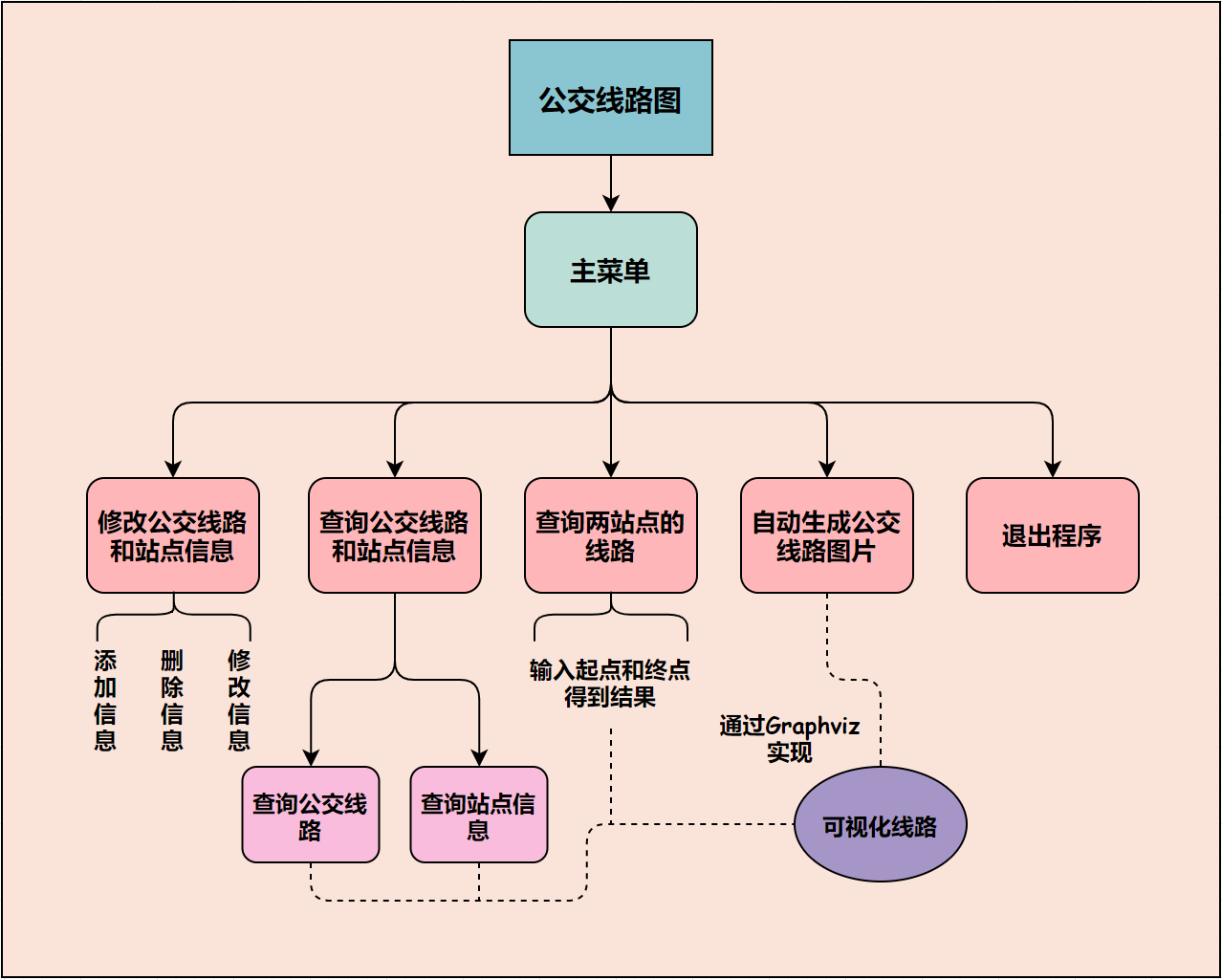


图1: 程序流程图



图2: 主界面菜单

**功能介绍：**

|  |  |
| --- | --- |
| 主菜单 | 通过图形化界面实现对公交线路图的功能的呈现。  主菜单分为5个功能：  1.修改公交线路和站点信息 2. 查询公交线路和站点信息  3.查询两站点线路 4.自动生成公交线路图 5.退出程序 |
| 修改公交线路和站点信息 | 通过利用文件和c++中的流操作，将文件中的内容先读取存到变量中，然后修改变量后，再重新写入到文件中。 |
| 查询公交线路和站点信息 | 通过文件中的数据信息建立公交线路图，然后利用dfs深度优先搜索和回溯算法查询每个公交的线路和经过某一个站点的所有线路，最后用动态数组保存每条路径结果。 |
| 查询两站点的线路 | 通过dfs深度优先搜索和两个变量来维护是否换乘，通过dfs找到所有至多换乘一次的线路，保存到动态数组中，然后输出。 |
| 自动生成公交线路图片 | 先通过c++中的文件操作，将线路图的数据信息写入到一个文件中，然后利用graphviz来创建一个dot文件，再通过c++自动调用cmd命令生成指定格式的图片。 |
| 退出程序 | 通过跳出主菜单的循环，使得程序退出。 |

## 1.设计特点

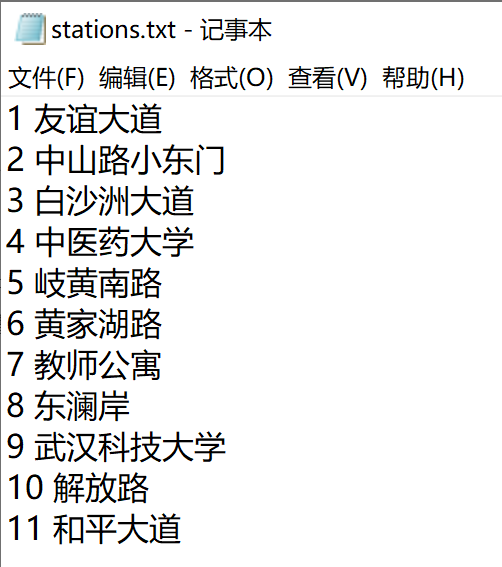
1. 程序采用结构化编程和面向对象编程技术，模块化实现程序，降低程序的耦合性。
2. 在创建公交线路图时采用链式前向星，使用链式前向星的优势是最大程度地减少内存占用，加快数据访问速度，此外本程序配合邻接矩阵来维护边权，使得程序更为方便地访问每条边的边权，两者相互结合，使得公交线路图的操作十分方便。
3. 在设计公交线路的查询操作时，该程序基于最短路的spfa和dijkstra的算法，采用深度优先搜索dfs和回溯进行维护每条公交线路，并用vector动态数组来储存公交车经过的线路，使得程序可以方便地访问每条路径。
4. 本程序采用图形化设计，所有界面和输入，输出均采用图形化设计。所以功能访问操作均由鼠标即可实现，利用easyx 实现界面框的设计和输入框的建立，同时实现了查询的结果显示，用户访问性十分友好。
5. 所有的公交线路图都实现了可视化设计，利用Graphviz 和系统自带的cmd 实现程序自动生成公交线路图，然后程序自动写入dot文件，最后自动转换为png图片格式，然后作为图片呈现到用户面前，所有操作均已完成自动化，使得公交线路图变得可视化。
6. 本程序配有点击边框和点击音效，使用户能够体验的应用化操作，没有程序与应用分离的感觉，并且界面的返回按钮，实现了应用的人性化。

## 2.程序整体结构及类的函数

|  |  |
| --- | --- |
| **Stations.h** | unordered\_map<string, int>mp;//名称与编号的映射  string name[N];  int cnt;//站点总数  void fun();//从文件中获取站点信息. |
| **Buses.h** | int tot;//tot表示公交车的数量.  vector<bus>buses;//用来储存公交车.  void fun();//从文件中获取公交车信息 |
| **Routes.h** | /\*  链式前向星建图.  无向边.  \*/  edge e[MAXN<<1];  int cnt,h[MAXN],vis[MAXN];  int g[MAXN][MAXN];//储存边权,便于直接访问.  //加边操作.  void add(int id, int u, int v, int w) {  g[u][v] = g[v][u] = w;  e[++cnt] = { v,h[u],w,id },h[u]=cnt;  //e[++cnt] = { u,h[v],w,id },h[v]=cnt;  }  void Create\_Graph();//创建图 |
| **FileOperation.h** | bool add\_station(string str1);//1.添加站点信息.  bool add\_bus(string str1);//2.添加公交车信息.  bool add\_route(string str1);//3.//添加线路信息.  bool delete\_station(string str1);//删除操作  bool delete\_bus(string str1);  bool delete\_route(string str1);  bool update\_station(string str1);//修改操作  bool update\_bus(string str1);  bool update\_route(string str1);  void show();//界面展示  void fun\_add();  void fun\_delete();  void fun\_update(); |
| **Function.h** | void dfs(int u, int fa,int st, int ed, vector<int>& path,int id);//搜索路径  void dfs1(int u, int fa, int st, int ed, vector<int>& path,int pre,int cnt,int sp);//搜索路径  void Init();//初始化公交线路图.  void YuChuLi();//获取公交车的线路路径并将路径上的点标记.  /\*  为验证公交线路图是否创建成功，程序需实现查询公交线路和站点信息功能。  Query\_1  (1) 查询公交线路  输入公交线路编号，系统通过公交线路编号查找到该线路途经的所有站点并输出。  (2) 查询站点信息  Query\_2  输入站点编号，系统通过站点编号查找到所有经过该站点的公交线路并输出。  Print\_Path 打印询问1,2的路径.  \*/  void Query\_1(int &jg);  void Query\_2();  void Print\_Path(int bh, int& yy);  /\*  3.查询两站点之间的路线，找到至多换乘1次的路线，并输出结果。  Query\_3  ok\_3 判断两站点是否连通.  Print\_st\_ed 打印该询问的路径.  \*/  int ok\_3;  void Query\_3();  void Print\_st\_ed(vector<int>& path,int sp,int& yy);//打印路径  void View\_Picture();//可视化路径  void View\_Path(vector<int>& path, int sp);  void View\_Path2(vector<int>& path);  void func2(); |
| Menu.h | void face();//展示菜单 |

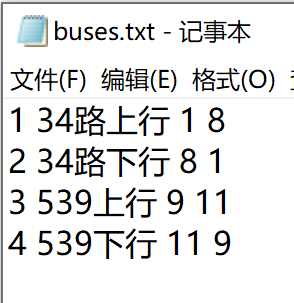
## 3.数据格式

**1.stations.txt**



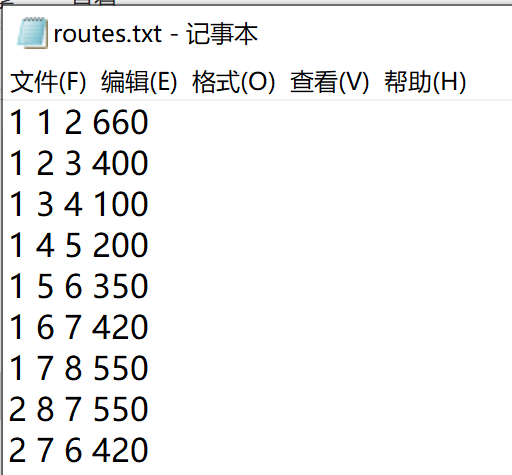
**站点信息由编号和站点名称组成。**

**2.buses.txt**



**公交车信息由编号、线路名、起点站编号、终点站编号组成。**

**3.routes.txt**



**线路由公交车编号、站点名、站点名、距离组成。**

## 4.数据结构（读文件创建图）

**1.公交车站点的储存和表示**

class Stations {

public:

unordered\_map<string, int>mp;

string name[N];

int cnt;//站点总数

void fun();//从文件中获取站点信息.

};

**通过map的字符串-整数 的映射，可以通过用站点名字来查找站点编号，通过字符串数组，可以通过编号来快速查找站点的名字。**

**2.公交车的储存和表示**

struct bus {

int id;//编号

string name;//线路名

int st, ed;//起点站编号,终点站编号.

//有参构造函数.

bus(int id,string name,int st,int ed):id(id),name(name),st(st),ed(ed){}

};

class Buses {

public:

int tot;//tot表示公交车的数量.

vector<bus>buses;//用来储存公交车.

void fun();

};

**公交车采用结构体的形式进行储存，包括了编号、线路名、起点站编号、终点站编号的属性，然后用类封装公交车类，用动态数组vector储存所有公交车。**

**3.公交线路图的建立和表示**

const int MAXN = 505;

struct edge {

int to, nt, w;

int id;

};

class Routes {

public:

/\*

链式前向星建图.

无向边.

\*/

edge e[MAXN<<1];

int cnt,h[MAXN],vis[MAXN];

int g[MAXN][MAXN];//储存边权,便于直接访问.

//加边操作.

void add(int id, int u, int v, int w) {

g[u][v] = g[v][u] = w;

e[++cnt] = { v,h[u],w,id },h[u]=cnt;

//e[++cnt] = { u,h[v],w,id },h[v]=cnt;

}

void Create\_Graph();

};

**通过链式前向星和邻接矩阵实现公交线路图的建立，建立图的储存结构，h数组表示每个结点的头结点，cnt表示边的个数。边用结构体封装，to表示连接的下一个结点，nt下一个边编号，w表示权值。**

**4.功能的数据结构实现和表示**

#pragma once

#include<cstdio>

#include "Buses.h"

/\*

实现公交车线路相关的功能.

\*/

#include "Routes.h"

#include "Stations.h"

using namespace std;

class Function {

public:

Buses b1;//公交车相关信息.

Routes r1;//线路图相关信息.

Stations s1;//站点信息.

vector<int>ans[505];

unordered\_map<int, int>mp[505],jg;

vector<vector<int> >res;

vector<int>resbj;

int vis[505];

void dfs(int u, int fa,int st, int ed, vector<int>& path,int id);

void dfs1(int u, int fa, int st, int ed, vector<int>& path,int pre,int cnt,int sp);

void Init();//初始化公交线路图.

void YuChuLi();//获取公交车的线路路径并将路径上的点标记.

/\*

为验证公交线路图是否创建成功，程序需实现查询公交线路和站点信息功能。

Query\_1

(1) 查询公交线路

输入公交线路编号，系统通过公交线路编号查找到该线路途经的所有站点并输出。

(2) 查询站点信息

Query\_2

输入站点编号，系统通过站点编号查找到所有经过该站点的公交线路并输出。

Print\_Path 打印询问1,2的路径.

\*/

void Query\_1(int &jg);

void Query\_2();

void Print\_Path(int bh, int& yy);

/\*

3.查询两站点之间的路线，找到至多换乘1次的路线，并输出结果。

Query\_3

ok\_3 判断两站点是否连通.

Print\_st\_ed 打印该询问的路径.

\*/

int ok\_3;

void Query\_3();

void Print\_st\_ed(vector<int>& path,int sp,int& yy);

void View\_Picture();

void View\_Path(vector<int>& path, int sp);

void View\_Path2(vector<int>& path);

void func2();

};

**Function功能类包括stations,buses,routes 其他三个类作为属性，vector<int>ans[i] 预处理出表示每个公交车对应的路径，mp用来储存每个公交车路径经过的结点，vis[]数组标记结点是否访问过。**

## 5.关键算法描述

**1.查询对应公交车经过的线路**

**dfs搜索第id辆公交车 从st->ed的路径,用path储存路径,最终保存到ans[id]中，利用链式前向星来遍历图的所有结点，如果该该节点为父亲结点或者访问过了就跳过continue,否则标记访问过，然后丢入path路径数组，然后继续访问下个结点，搜索完该路径后，回溯标记vis[v]=0，然后该节点push\_back丢出数组，实现深度优先搜索实现搜索所有公交车对应的路径。**

void Function::dfs(int u, int fa,int st, int ed, vector<int>& path,int id) {

//dfs搜索第id辆公交车 从st->ed的路径,用path储存路径,最终保存到ans[id]中.

//printf("u=%d,fa=%d\n", u, fa);

if (u == ed) {

ans[id] = path;

return;

}

//printf("---%d\n", r1.h[u]);

for (int i = r1.h[u]; i; i = r1.e[i].nt) {

int v = r1.e[i].to;

//printf("%d %d %d %d\n",i,r1.e[i].to,r1.e[i].nt,r1.e[i].w);

if (v == fa ||vis[v]) continue;

vis[v] = 1;

path.pb(v);

dfs(v, u, st, ed, path,id);

vis[v] = 0;

path.pop\_back();

}

}

**2.查询最多换乘一次的线路**

**dfs搜索st->ed是否存在至多换乘一次的路径,并输出路径.**

**pre表示前一条边的所属线路,-1表示从起点开始.**

**cnt表示已经换乘的次数.**

**sp表示当前换乘的边的目标站点.**

**每次访问结点时，特判4种情况 是否换乘，来处理，如果已经换乘了一次且此时需要换乘则跳过，否则可以换乘然后继续搜索。**

void Function::dfs1(int u, int fa, int st, int ed ,vector<int>& path,int pre,int cnt,int sp) {

/\*dfs搜索st->ed是否存在至多换乘一次的路径,并输出路径.

pre表示前一条边的所属线路,-1表示从起点开始.

cnt表示已经换乘的次数.

sp表示当前换乘的边的目标站点.

\*/

//closegraph();

if (u == ed) {

ok\_3 = 1;

res.push\_back(path);

resbj.push\_back(sp);

//Print\_st\_ed(path,sp);

return;

}

for (int i = r1.h[u]; i; i = r1.e[i].nt) {

int v = r1.e[i].to, id = r1.e[i].id;

printf("%d %d pre=%d, id=%d vis[%d]=%d\n", u, v,pre,id,v,vis[v]);

if (v == fa || vis[v]) continue;

if (pre!=-1&&pre!=id &&cnt==1) continue;//如果已经换乘了一次且当前又要换乘.

int ppre, cntt,spp;

if (pre != -1 && pre != id && cnt == 0) { //如果还没有换乘过,且要换乘.

ppre = id, cntt = 1,spp= v;

}

else if (pre == -1) { //如果从起点开始.

ppre = id, cntt = 0;

if (~sp) spp = sp;

else spp = -1;

}

else ppre = id, cntt = cnt,spp=(sp!=-1)?sp:-1;

vis[v] = 1;

path.pb(v);

dfs1(v, u, st, ed, path, ppre,cntt,spp);

vis[v] = 0;

path.pop\_back();

}

}

**3.打印路径的算法实现**

**利用之前搜索的结果，将sp标记传入是否换乘，然后创建dot文件写入对应的线路，找到换乘在图中对应的边，用红色边特殊标记。**

**可视化实现线路显示。**

void Function::Print\_st\_ed(vector<int>& path,int sp,int &yy) {

/\*

sp表示换乘的边的站点.

\*/

//printf("sp=%d\n", sp);

/\*

利用Graphviz　实现可视化!!!!!!!

\*/

ofstream os("my.dot");

os << "digraph G {" << '\n';//创建一个有向图.

os << " rankdir = LR" << '\n';

os << " node [fontname=\"Microsoft YaHei\" charset=\"UTF-8\"]" << '\n';

os << " edge [color=red]" << '\n';

//printf("路径为:\n");

//printf("%d", path[0]);

int pre=-1;//记录 要换乘的起始边.

for (int i = 1; i < path.size(); i++) {

//printf(" -> %d", path[i]);

//os << " "<< path[i - 1] << " -> " << path[i] << '\n';

if (path[i] == sp) pre = path[i - 1];

}

//printf("\n");

//cout << s1.name[path[0]];

int xx = 50;

outtextxy(xx, yy, s1.name[path[0]].c\_str());

int delta = textwidth(s1.name[path[0]].c\_str());

xx += delta;

string str1 = " -> ";

for (int i = 1; i < path.size(); i++) {

string str2 = str1 + s1.name[path[i]];

delta = textwidth(str2.c\_str());

//xx += delta;

if (xx + textwidth(str2.c\_str()) > 630) {

xx = 50;

yy += 30;

}

outtextxy(xx, yy, str2.c\_str());

xx += delta;

//cout << " -> " << s1.name[path[i]];

os << " \"" << string\_To\_UTF8(s1.name[path[i - 1]]) << "\" -> \"" << string\_To\_UTF8(s1.name[path[i]])<< "\" ";

//if (sp == path[i]) cout << "[label=\"需要换乘\"]";

//cout<<"\n";

}

os << "}" << '\n';

os.close();

system("dot -Tpng my.dot -o my.png");

if (~sp) {

str1 = " (需要换乘一次)";

if (xx + textwidth(str1.c\_str()) > 640) {

xx = 50;

yy += 30;

}

outtextxy(xx, yy, str1.c\_str());

}

else {

str1 = " (不需要换乘)";

if (xx + textwidth(str1.c\_str()) > 640) {

xx = 50;

yy += 30;

}

outtextxy(xx, yy, str1.c\_str());

}

/\*printf("\n");

if (sp != -1) {

printf("%d -> %d 要换乘\n", pre, sp);

cout << s1.name[pre] << " -> " << s1.name[sp] << " 要换乘" << endl;

}

else {

printf("该线路不需要换乘\n");

}

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

\*/

}

**4.修改文件的具体算法实现(添加信息举例，其他类似)**

**打开对应信息的文件，然后用getline读入每行信息，用sscanf\_s将字符串格式化读入，然后判断，修改对应的信息。**

bool FileOperation::add\_station(string str1) {

//1.添加站点信息.

int ok = 1;

ifstream fi;

fi.open("stations.txt");

int cnt = 1;

string str;

int bh = 0;

while (getline(fi, str)) {

int num = 0;

string x;

char a[50] = { 0 };

sscanf\_s(str.c\_str(), "%d %s", &num, a, 50);

//str = str.substr(i + 1);

//cout << str << " " << num << endl;

x = a;

if (x == str1) {

ok = 0;

}

cnt++;

}

fi.close();

if (!ok) return false;

else {

ofstream of;

of.open("stations.txt", ios::app);

of << cnt << " " << str1 << '\n';

of.close();

return true;

}

}

**5.图形化界面的算法实现**

**利用easyx 进行绘图，利用while循环+mousemsg 来实现鼠标点击对应的功能，当鼠标移动到对应的功能区域并按下左键后，就执行相应的功能，然后跳出while循环。**

void Menu::face() {

const char\* str[5] = {

"1.修改公交线路和站点信息",

"2.查询公交线路和站点信息",

"3.查询两站点之间的路线",

"4.自动生成公交线路图片",

"5.退出程序"

};

loop:

MOUSEMSG m;

// 初始化绘图窗口

initgraph(640, 480);

IMAGE background;//定义一个图片名.

loadimage(&background, "bk1.jpg", 640, 480, 1);//从图片文件获取图像

putimage(0, 0, &background);//绘制图像到屏幕，图片左上角坐标为(0,0)

setfillcolor(RED);

//fillrectangle(100, 30, 500, 80);

int lx = 160, rx = 450, uy = 100, dy = 140, delta = 70;

for (int i = 0; i < 5; i++) { //填充矩形.

fillrectangle(lx, uy, rx, dy);

uy += delta, dy += delta;

}

settextstyle(30, 0, "宋体");

setbkmode(TRANSPARENT);

settextcolor(BLACK);

outtextxy(230, 45, "公交线路图");

settextstyle(18, 0, "Consolas");

settextcolor(WHITE);

int str\_x = 210, str\_y = 110;

for (int i = 0; i < 5; i++) { //输出字符串

outtextxy(str\_x, str\_y, str[i]);

str\_y += delta;

}

bool over = 0;

int y1 = 95, y2 = 150;

FileOperation fo;

Function fc;

fc.Init();

while (1) {

m = GetMouseMsg();

if (m.x >= 160 && m.x <= 450 && m.y >= 100 && m.y <= 140) {

setlinecolor(BLUE);

rectangle(150, y1, 460, y2);

if (m.uMsg == WM\_LBUTTONDOWN) {

PlaySound("Appclick.wav", NULL, SND\_FILENAME | SND\_ASYNC);

closegraph();

fo.show();

// fo.show(); //修改公交线路和站点信息.

// func2(fc);

// fc.Query\_2();

break;

}

}

else if (m.x >= 160 && m.x <= 450 && m.y >= 170 && m.y <= 210) {

setlinecolor(BLUE);

rectangle(150, y1 + delta, 460, y2 + delta);

if (m.uMsg == WM\_LBUTTONDOWN) {

PlaySound("Appclick.wav", NULL, SND\_FILENAME | SND\_ASYNC);

closegraph(); //查询公交线路和站点信息

fc.func2();

// fc.Query\_1();

break;

}

}

else if (m.x >= 160 && m.x <= 450 && m.y >= 240 && m.y <= 280) {

setlinecolor(BLUE);

rectangle(150, y1 + delta \* 2, 460, y2 + delta \* 2);

if (m.uMsg == WM\_LBUTTONDOWN) {

PlaySound("Appclick.wav", NULL, SND\_FILENAME | SND\_ASYNC);

closegraph(); //查询两站点之间的路线

fc.Query\_3();

break;

}

}

else if (m.x >= 160 && m.x <= 450 && m.y >= 310 && m.y <= 350) {

setlinecolor(BLUE);

rectangle(150, y1 + delta \* 3, 460, y2 + delta \* 3);

if (m.uMsg == WM\_LBUTTONDOWN) {

PlaySound("Appclick.wav", NULL, SND\_FILENAME | SND\_ASYNC);

closegraph(); //查询两站点之间的路线

fc.View\_Picture();

break;

}

}

else if (m.x >= 160 && m.x <= 450 && m.y >= 380 && m.y <= 430) {

setlinecolor(BLUE);

rectangle(150, y1 + delta \* 4, 460, y2 + delta \* 4);

if (m.uMsg == WM\_LBUTTONDOWN) {

PlaySound("Appclick.wav", NULL, SND\_FILENAME | SND\_ASYNC);

closegraph(); //6.退出程序

//system("cls");

return;

}

}

else {

setlinecolor(WHITE);

//rectangle(95, 25, 510, 90);

rectangle(150, y1, 460, y2);

rectangle(150, y1 + delta, 460, y2 + delta);

rectangle(150, y1 + delta \* 2, 460, y2 + delta \* 2);

rectangle(150, y1 + delta \* 3, 460, y2 + delta \* 3);

rectangle(150, y1 + delta \* 4, 460, y2 + delta \* 4);

}

}

goto loop;

// 按任意键退出

}

## 6.查询公交线路和站点信息

为验证公交线路图是否创建成功，程序需实现查询公交线路和站点信息功能。

(1) 查询公交线路

输入公交线路编号，系统通过公交线路编号查找到该线路途经的所有站点并输出。

**输入公交线路编号，如下图1**

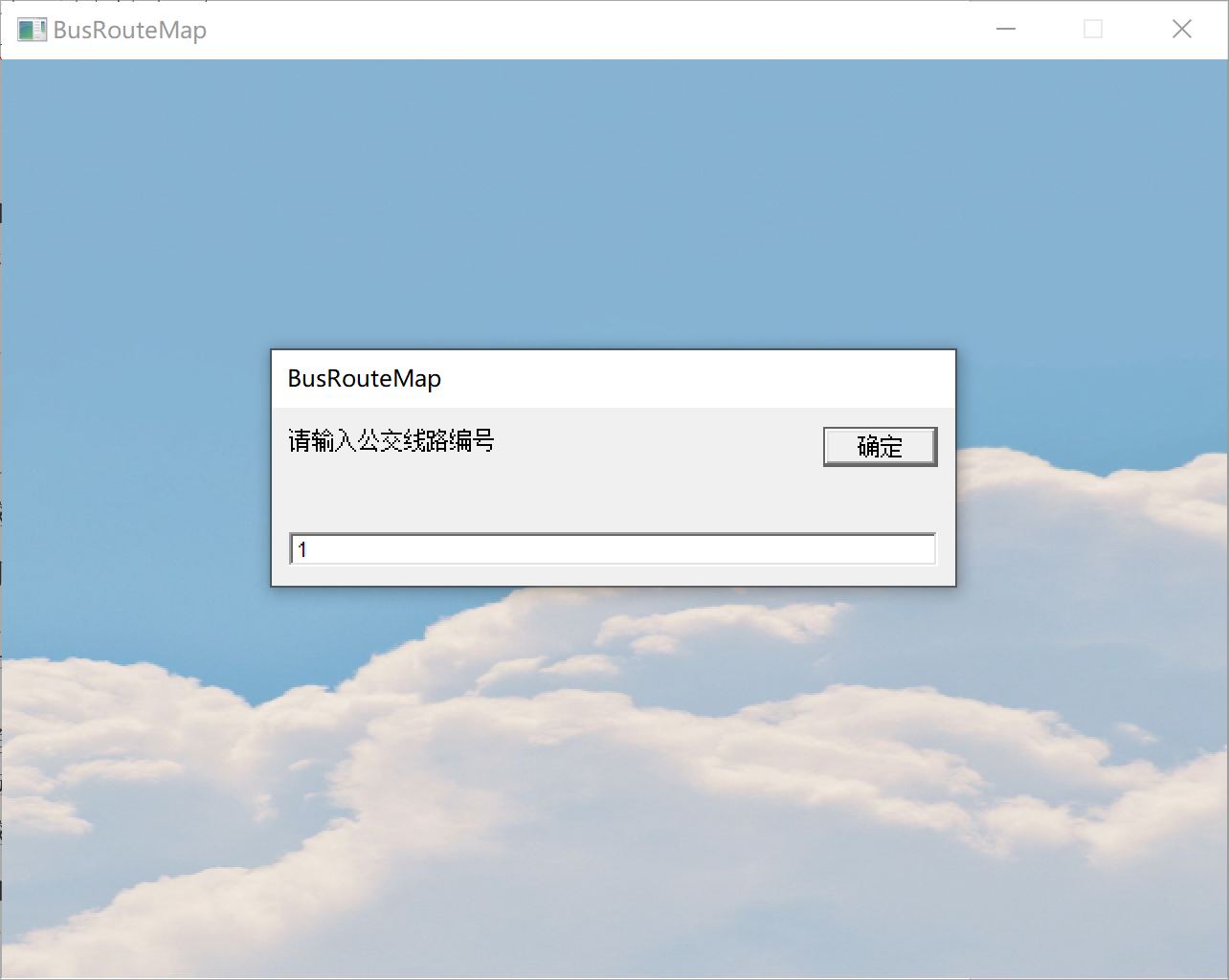


图1 公交车编号界面

**会得到该公交车对应的线路，还可选择是否可视化该线路，如下图2.**

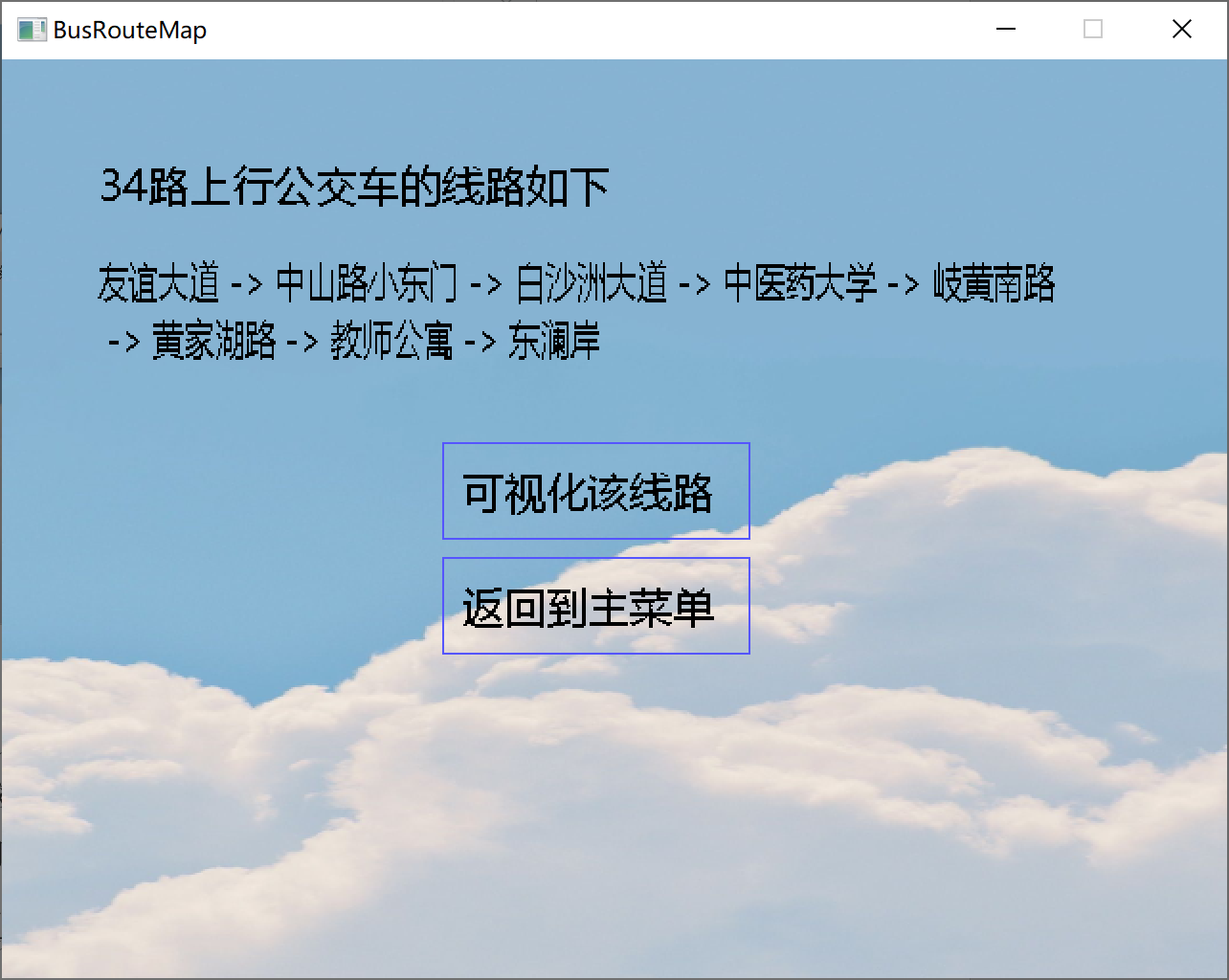


图2 显示线路

**(2) 查询站点信息**

**输入站点编号，系统通过站点编号查找到所有经过该站点的公交线路并输出。**

**输入站点编号，如下图3.**

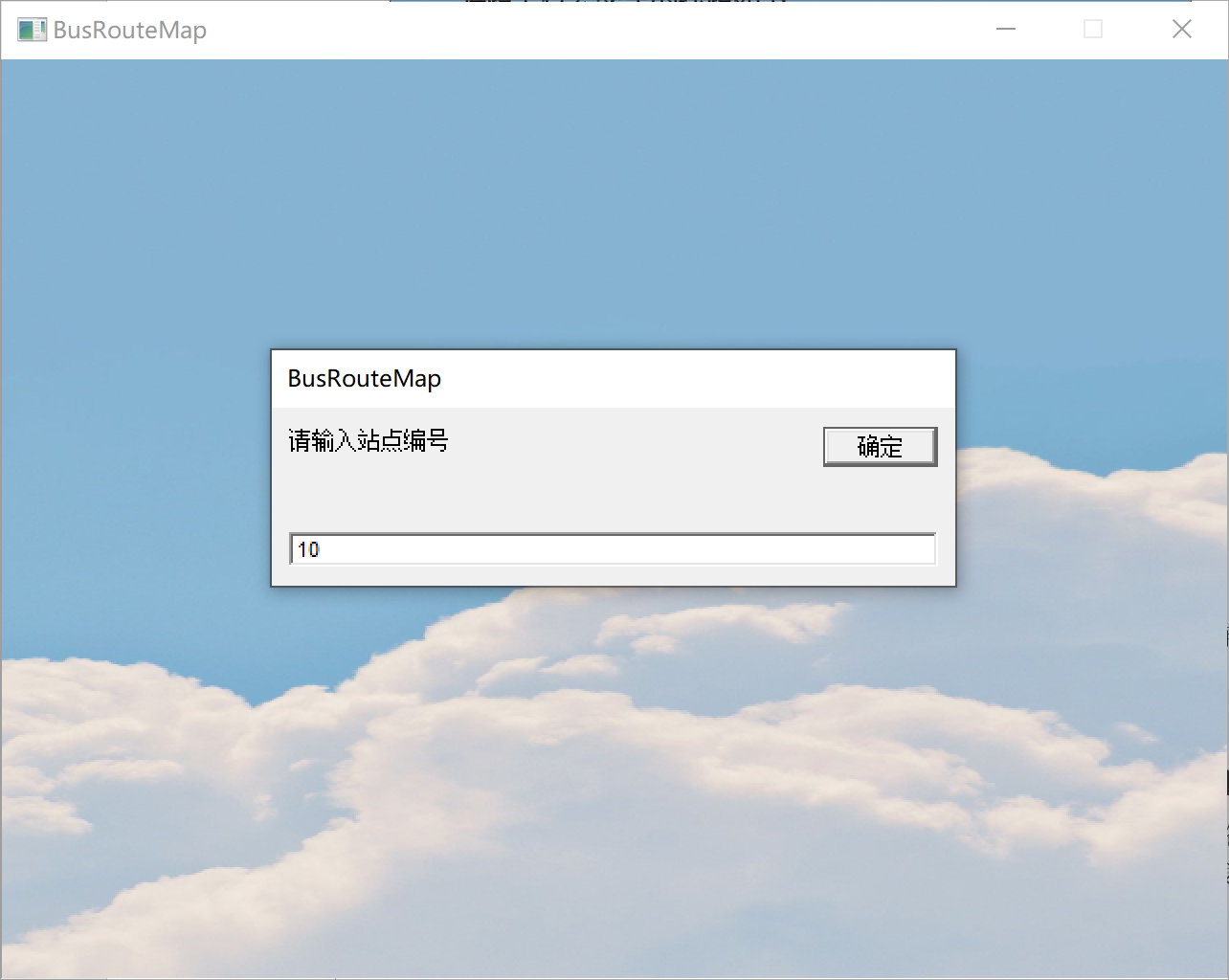


图3 站点编号

**程序会输出经过该站点的所有线路，如下图4.**

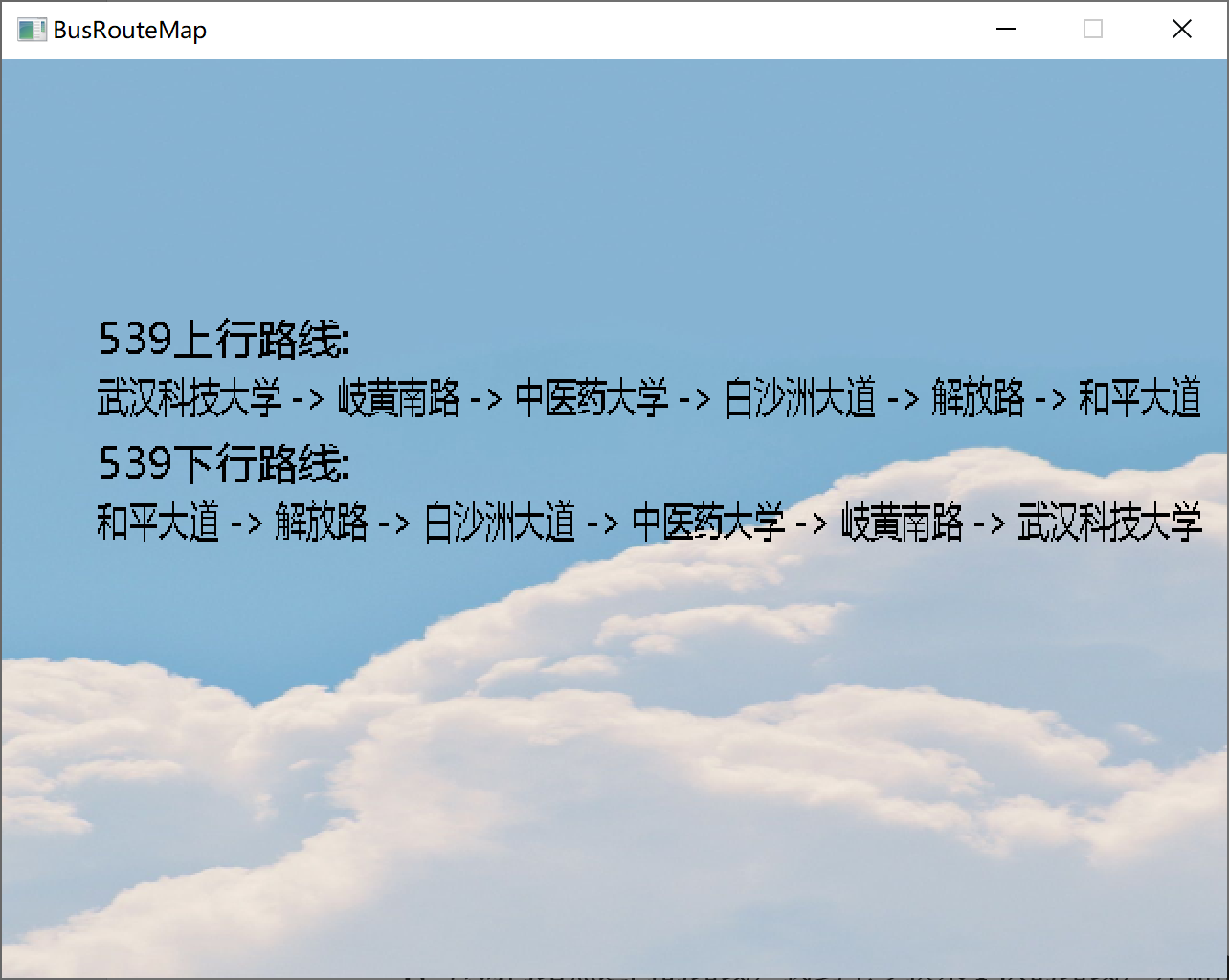


图4 显示站点线路

## 7.查询两站点之间的路线，找到至多换乘1次的路线，并输出结果。

**用户输入要查询的起点和终点，程序将先判断两个站点之间是否有一条路径（即两个顶点之间是否连通）。若两个站点之间有路线，则找到所有最多换乘1次的路线，然后依次输出。**

**输入要查询的起点和终点，如下图1。**

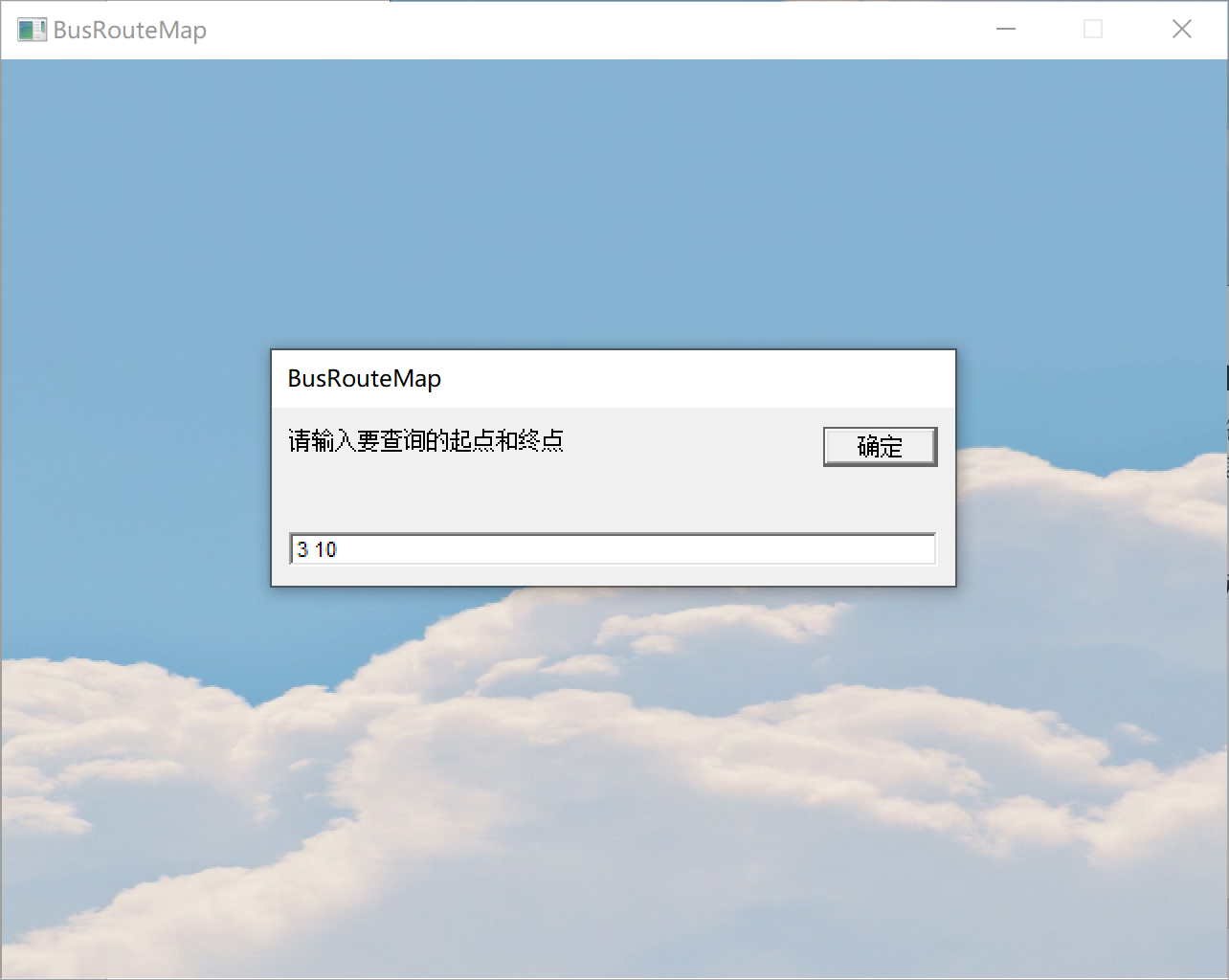


图1 查询路线

**程序会输出从起点到终点至多换乘一次的所有路线，还可选择可视化任意一条路线，如下图2。**

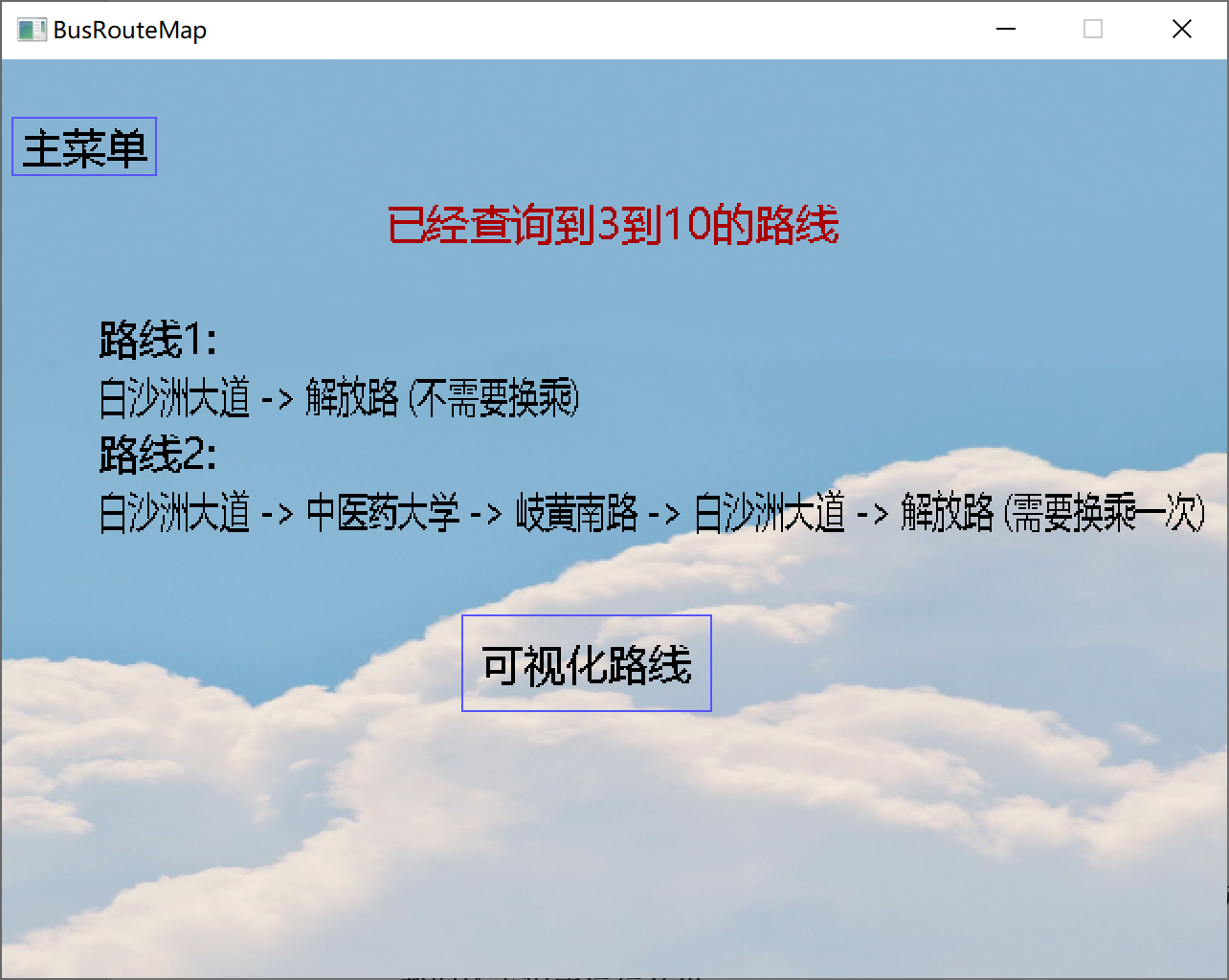


图2 显示线路

**可视化路线2如下图3**

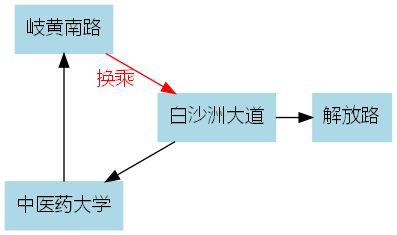


图3

## 8.修改公交线路和站点信息，保存文件

**修改公交线路和站点信息的界面，如下图4**

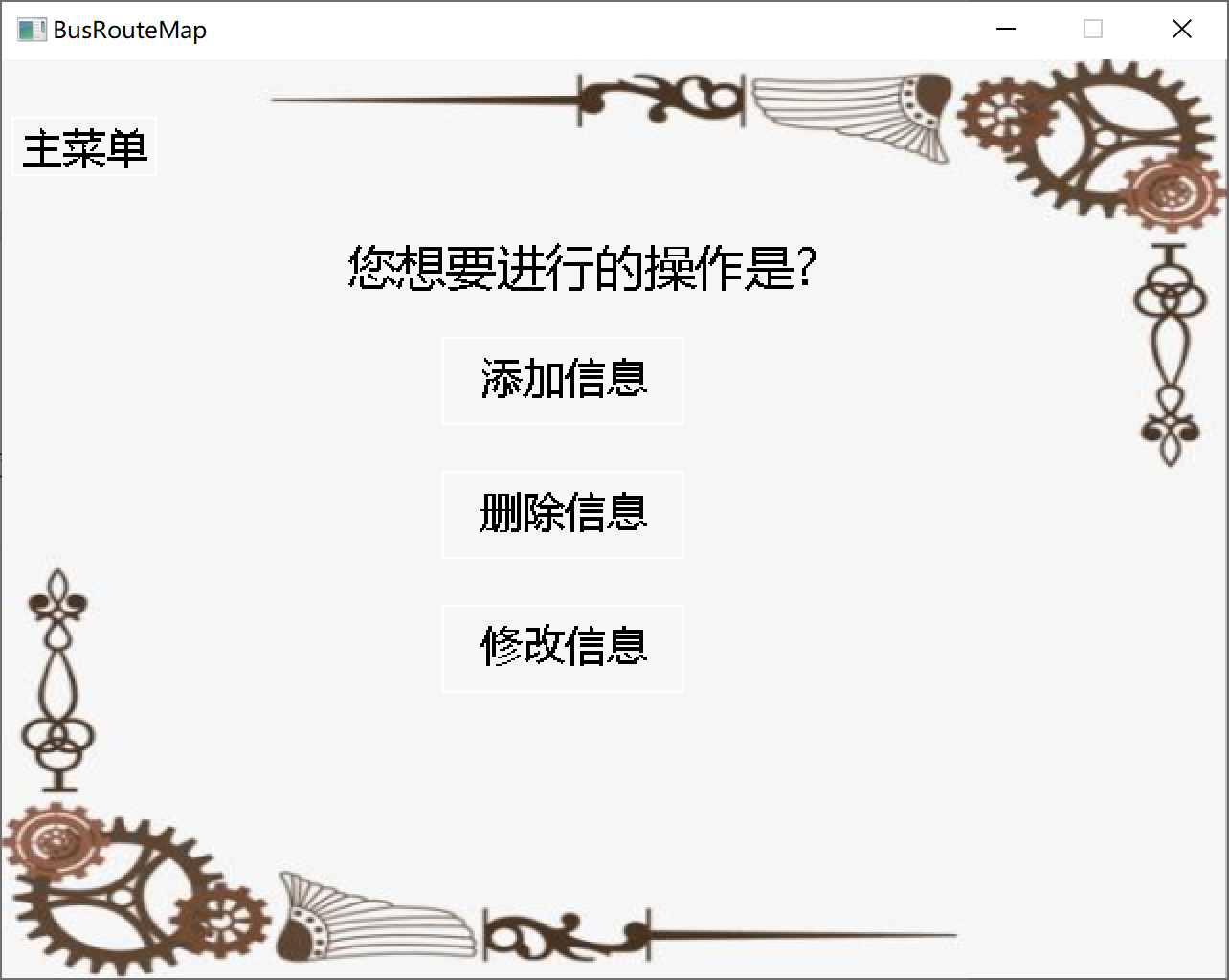


图4 信息操作界面

**添加公交线路，如下图5**



图5 添加操作界面

**输入要添加的站点，如下图6**

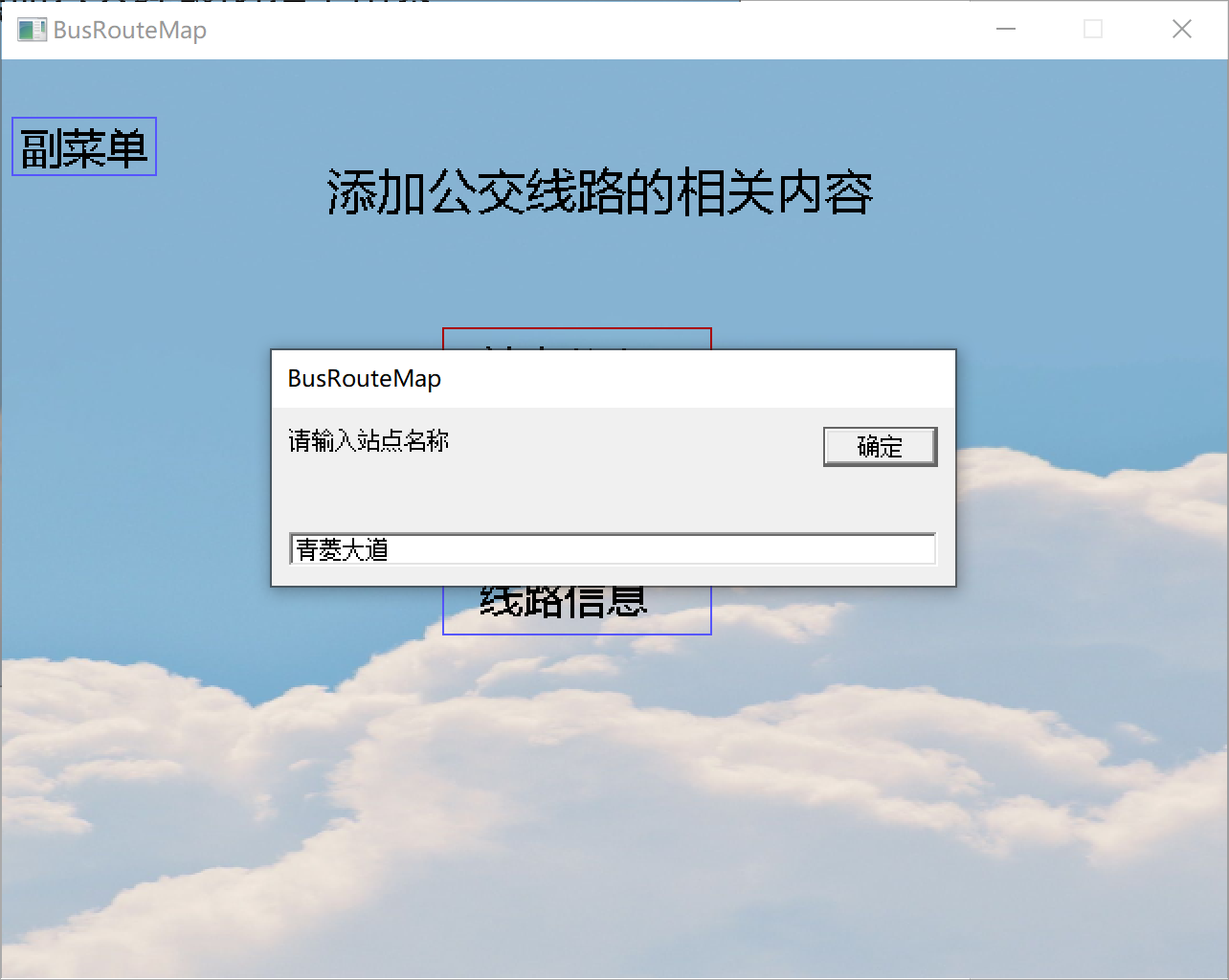


图6

**程序会将内容储存到文件中，如下图7**

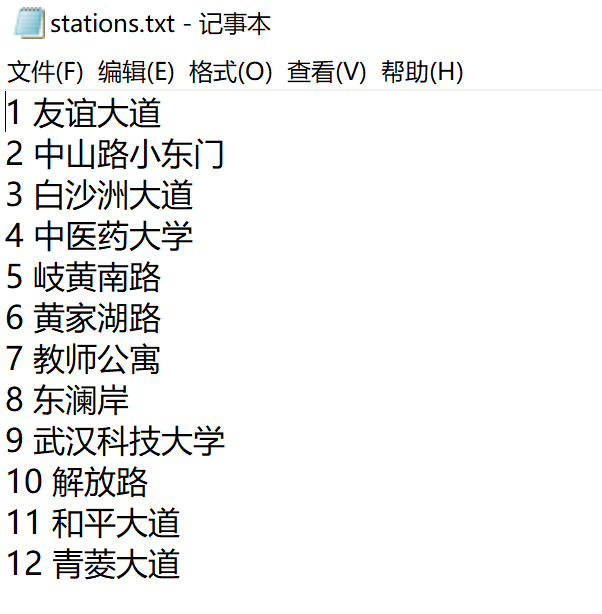


图7

**删除和修改操作类似，可以通过该操作增加，修改，删除站点信息，公交车信息和公交线路图的信息。**

## 9.设计总结

通过本次数据结构试验，我对于利用数据结构来设计程序有了更深刻的理解和认知，通过设计公交线路图，使我对图论的知识了解了更多，学会了如何利用链式前向星和邻接矩阵建图，掌握了利用深度优先搜索和宽度优先搜索进行程序设计，同时我学会如何进行程序的可视化，将公交线路图进行可视化，通过图片的形式程序，实现了代码和图片的转换。并且熟练掌握了文件的相关操作，学到了许多关于文件的增删改查的知识，提高了我对于计算机内部的文件操作的理解能力，更好地锻炼了我解决问题的能力。同时本次实验很好地锻炼了我分块化设计和动手操作的技能，让我明白面对一个项目如何进行思考和进行系统化设计。总而言之，通过本次实验，我学习和收获了许多课本上难得获取的知识和能力，让我对计算机的学习有了新的思考和方向。