

華中科技大學

# 课程实验报告

课程名称：C++程序设计

实验名称：面向对象的公交地图导航

院    系：计算机科学与技术

专业班级：CS2008 班

学    号：U202015533

姓    名：徐瑞达

指导教师：许向文

2021 年 11 月 29 日

## 一、需求分析

### 1. 题目要求

假定所有公交车辆从起点到终点都是双向非环路的，且双向线路的所有停靠站点都对应相同。设有  $M$  路公交车，第  $j$  路公交车有  $N_j$  个站点。所有公交线路累计共有  $S$  个站点，第  $k$  个站点的坐标为  $(X_k, Y_k)$ ，所有坐标均以米为单位标注。邻近站点之间的距离指的是站点坐标之间的欧几里得距离。现有一人处于起点坐标  $(X_b, Y_b)$ ，此人需要步行到最近站点乘车，下车后要步行到达的终点坐标为  $(X_e, Y_e)$ ，而他特别不愿意走路，能坐公交就尽量坐公交。假定公交转乘时的步行距离为 0，试编程求他从起点  $(X_b, Y_b)$  到终点  $(X_e, Y_e)$  的转乘次数最少的乘坐线路。

所有公交线路的站点坐标存放于“stops.txt”文件，其中第 1 行为站点总个数，第 2 行为第 1 个站点的坐标，第 3 行为第 2 个站点的坐标，以此类推。可用图形化的界面显示站点及公交线路。

所有公交线路信息存放于“lines.txt”文件，其中第 1 行为公交线路总数，第 2 行为每条公交线路的站点总数，第 3 行为线路 1 经过的站点编号（对应站点坐标参见“stops.txt”），第 4 行为线路 2 经过的站点编号，以此类推。

通过类型抽象形成站点、公交线路、转乘站点、转乘线路、转乘矩阵、公交系统等类，输入上述文件初始化站点和线路对象，然后通过图形化的界面显示站点和线路地图。用户用鼠标左键在地图上设定起点、鼠标右键确定终点，按照设定的最少转乘或者最短距离选项规划出从起点步行到最近站点上车、到离终点最近站点下车步行到终点的线路，在地图上用不同颜色显示规划线路若干秒，然后消除并恢复原始地图线路的颜色。

用户可输入“华科大”或“华中科技大学”查询其所在位置，通过最大公共子串算法进行模糊匹配，找到“华中科技大学”及其坐标。在确定始发单位坐标和终点单位坐标后，按照设定的最少转乘或者最短距离选项规划线路，并在地图上用不同颜色显示规划线路若干秒，然后消除并恢复原始地图线路的颜色。

### 2. 需求分析

- (1) 定义合适的类以存储站点、公交线路、转乘路径等数据信息；
- (2) 定义合适的类以实现闭包运算，完成站点间转乘路径的方案查找；
- (3) 将 Qt 图形化界面与所定义的类结合，以实现文件读取，选取出发和到达位置，模糊查询机构位置，规划路线并提示最短路径等功能；

(4) 图形化界面应有良好的操作体验，应有使用提示和快捷键操作，便于用户操作。

## 二、系统设计

### 1. 概要设计

(1) 系统框架和结构：系统按照功能分为 4 个模块，分别为文件的读取和处理，数据的图形化显示，用户交互选择和方案的决策与图形化显示，其流程图如 5-1 所示：

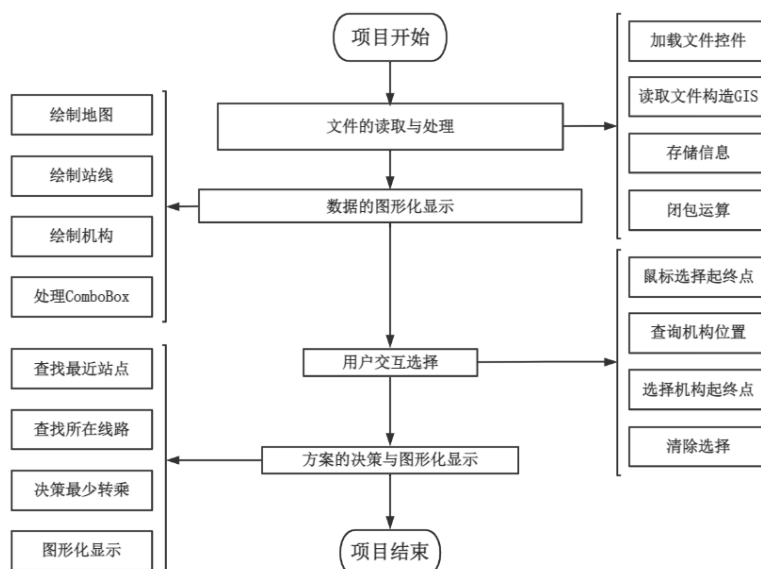


图 5-1 系统框架与结构

(2) 用户操作流程：用户操作流程为加载文件，选择起终点/查询机构位置，继续选择/查询，退出程序。如图 5-2 所示：

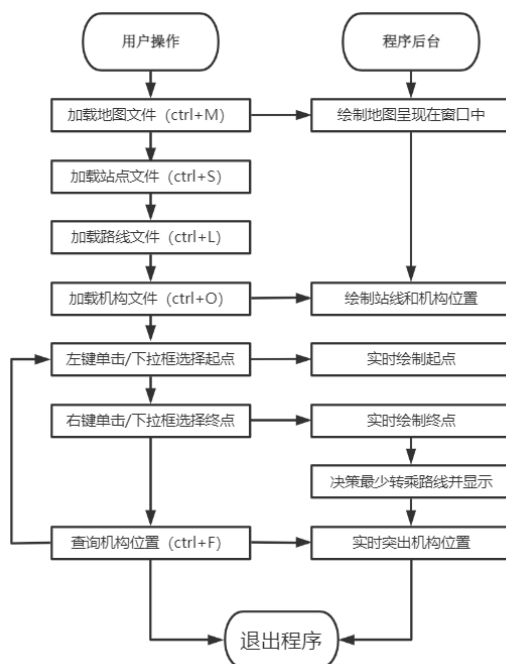


图 5-2 用户操作流程

## 2. 详细设计

### (1) 文件的读取和处理:

I 为方便用户操作，理想的文件加载方式为通过 Qt 菜单栏控件/快捷键选择文件。具体实现方式为使用 `QFileDialog` 打开子窗口选择文件。读取地图文件后，将立即调用 `update` 实现重绘事件，以将地图绘制在窗口中。读取站点、路线、机构文件后，调用 `GIS` 类的构造函数构造 `gis` 实例类。首先将站点、路线、机构信息读取至相应的 `stop`、`line`、`organization` 数组中，然后初始化原始闭包并进行闭包乘法计算直至任意两条不同的线路均有转乘方案可选择。完成 `gis` 实例类构造后，立即调用 `update` 实现重绘事件，将站点、机构位置、公交线路（默认不显示，用户可选择显示公交线路）绘制在地图上，同时更新查询子窗口、起点和终点的 `QcomboBox` 的列表信息。

II 涉及到的具体函数和数据结构分析:

- `GIS::GIS(const char* flstop, const char* flline, const char* florg)`

该函数为 `GIS` 类的构造函数，参数为三个存储文件路径的字符串。在函数中，首先读取站点文件到 `GIS` 的数据成员 `st`，然后读取路线文件到 `GIS` 的数据成员 `ls` 中，接着读取机构文件并进行模糊查找处理；接下来初始化原始闭包 `raw`，对于每两条线路，寻找转乘站点并添加到对应路径方案中，初始化完成后，进行闭包累乘运算，同时实时检查是否仍有两条线路无转乘方案，如果已经全部找到至少一条转乘路径，则停止闭包运算，完成转成路径的查找。

- 该过程中用到的辅助数据结构有如下:

`typedef struct filename{//存储有关文件路径等信息的结构体`

`QString map;`

`QString stops;`

`QString lines;`

`QString orgs;`

`int newx,newy;`

`int MapFlag,StopFlag,LineFlag,OrgFlag,LineFlagPaint;`

`}FilePath;`

`typedef struct org{//描述机构信息`

```
char *name;

int x,y;

int order;

}ORG;

class STOP {          //描述一个公交站点
    int number;        //所有公交站点编号
    int x, y;          //公交站点坐标
public:
    STOP(int n = 0, int x = 0, int y = 0);
    virtual int& X();
    virtual int& Y();
    virtual int& N();
};

class LINE {          //描述一条公交线路
    const int numb; //公交线路编号从 1 开始
    int* const stop; //公交线路上所有站点编号
    const int nofs; //公交线路上站点数量
public:
    LINE(int numb = 0, int nofs = 0, int* stop = nullptr);
    LINE(const LINE& r);
    LINE(LINE&& r) noexcept;
    LINE& operator=(const LINE& r);
    LINE& operator=(LINE&& r)noexcept;
    virtual int has(int s)const;    //线路是否包含站点编号 s,返回线路中的站次序号:
    -1 表示没有

    virtual int cross(const LINE& b)const; //两条公交线路相交则返回 1
    virtual operator int()const;    //取公交线路编号
    virtual int NOFS()const;        //取公交线路的站点数量
    virtual double dist(int d, int a)const; //线路从站次 d 到站次 e 的距离
    virtual int& operator[](int x);//取线路某个站次的站点编号
```

```
virtual ~LINE()noexcept;

};

class TRAN {          //从线路 from 经站点编号 stop 转至线路 to
    int from;          //现在乘坐的公交线路号
    int to;            //需要转乘的公交线路号
    int stop;          //由 stops.txt 定义的站点编号
public:
    TRAN(int from = 0, int to = 0, int stop = 0);
    int operator==(const TRAN& t)const;//判断两转乘是否相等
    virtual int& F();//现在乘坐的公交线路号
    virtual int& T();//需要转乘的公交线路号
    virtual int& S();//转乘点的站点编号
};

class ROUTE {          //一条转乘路径
    TRAN* const tran;//转乘路径上的所有转乘
    const int noft; //转乘路径上转乘次数
public:
    ROUTE(TRAN* tran = nullptr, int noft = 0);
    ROUTE(const TRAN& t);
    ROUTE(const ROUTE& r);
    ROUTE(ROUTE&& r) noexcept;
    virtual int print()const;
    virtual operator int()const;          //得到转乘次数
    virtual int operator==(const ROUTE& r)const;//判断两转乘路径是有相同
    virtual ROUTE operator *()const;      //去重复公交转乘
    virtual TRAN& operator[](int);        //返回某个转乘节点
    virtual ROUTE operator+(const ROUTE& r)const;
```

```

virtual ROUTE& operator=(const ROUTE& r);
virtual ROUTE& operator=(ROUTE&& r) noexcept;
virtual ROUTE& operator+=(const ROUTE& r);
virtual ~ROUTE() noexcept;
};

class NODE {          //闭包矩阵元素：记载的转乘次数和线路
    ROUTE* const p; //闭包矩阵 r*c 个元素记载的转乘路径方案
    int n;          //闭包矩阵 r*c 个元素记载的转乘路径方案数
public:
    NODE(ROUTE* p, int n);
    NODE(int n = 0);
    NODE(const NODE& n);
    NODE(NODE&& n)noexcept;
    virtual NODE  operator*()const; //矩阵元素约简：去掉转乘中的环
    virtual NODE  operator+(const ROUTE& n)const;
    virtual NODE  operator+(const NODE& n)const;
    virtual NODE  operator*(const NODE& n)const;
    virtual NODE& operator=(const NODE& n);
    virtual NODE& operator+=(const NODE& n);
    virtual NODE& operator+=(const ROUTE& n);
    virtual NODE& operator*=(const NODE& n);
    virtual NODE& operator=(NODE&& n)noexcept;
    virtual ROUTE& operator [] (int x);
    virtual operator int& ();          //可转乘路径数 n
    virtual ~NODE()noexcept;
    virtual void print()const;        //打印转乘矩阵的元素
};

class TMAP {          //所有公交转乘元素的闭包矩阵

```

```

    NODE* const p; //指向闭包矩阵的 r*c 个元素
    const int r, c; //闭包矩阵的行数和列数
public:
    TMAP(int r = 0, int c = 0);
    TMAP(const TMAP& a);
    TMAP(TMAP&& a)noexcept;
    virtual ~TMAP();
    virtual int notZero()const; //若有不可达站点则返回 0
    virtual ROUTE miniTran(int b, int e)const; //起点站次 b,终点站次 e,返回最小转
乘路径
    //int getNUM(int b, int e, ROUTE& r); //起点站次 b,终点站次 e,基于路径 r 的转乘
次数
    virtual NODE* operator[](int r); //得到存储多种路径的 NODE 类元
素的某行首址
    virtual int& operator()(int r, int c); //得到 r 到 c 可转乘路径数目
    virtual TMAP operator*(const TMAP& a)const; //闭包运算：乘法
    virtual TMAP operator+(const TMAP& a)const; //闭包运算：加法
    virtual TMAP& operator=(const TMAP& a);
    virtual TMAP& operator=(TMAP&& a)noexcept;
    virtual TMAP& operator+=(const TMAP& a);
    virtual TMAP& operator*=(const TMAP& a);
    virtual TMAP& operator()(int r, int c, const ROUTE& a); //将路径加入到 r 行 c 列元
素中
    virtual void print()const; //打印转乘矩阵
};
typedef struct answer{//存储得到的最小转乘路径的结构体
    int f,t,n,flag;//f,t 指出发和下车站点， n 指转乘次数， flag 标记是否已经得到结果
    ROUTE r[100];
}ANS;
struct GIS { //描述地理信息系统的类

```



```
public:
    static STOP* st;          //所有公交站点
    static LINE* ls;          //所有公交线路
    static int  ns, nl, no;    //公交站数 ns, 公交线路数 nl
    static TMAP raw, tra;     //原始转乘矩阵, 闭包转乘矩阵
    static ORG *org;          //所有机构信息
    GIS();
    GIS(const char* flstop, const char* flline, const char* florg); //根据文件路径加载信息
    void print()const;
    int* nearstop(int x, int y); //寻找距离(x,y)最近的站点并存入 int 数组中返回
    void miniTran(CHOICE *from, CHOICE *to, ANS *ans); //根据出发点和终点坐标,
    以及是否在车站处寻找最小转乘路径
    ~GIS();
};
```

## (2) 数据的图形化显示:

I 软件布局与逻辑: 软件布局如图 5-3 所示



图 5-3 软件布局图

其中菜单栏中分为三栏, 文件菜单栏用于加载文件, 查询菜单栏可进入查询机构系统, 帮助菜单栏可查看软件使用帮助, 快捷键指南, 版权信息; 菜单栏下方中, 起点和终点栏可选择机构, 并可根据鼠标点击事件提示起点和终点坐标, 同时提供清除选择与显示/隐藏公交线路按钮。

II 加载成功后的默认显示与显示路线界面: 如图 5-4-1 与 5-4-2 所示

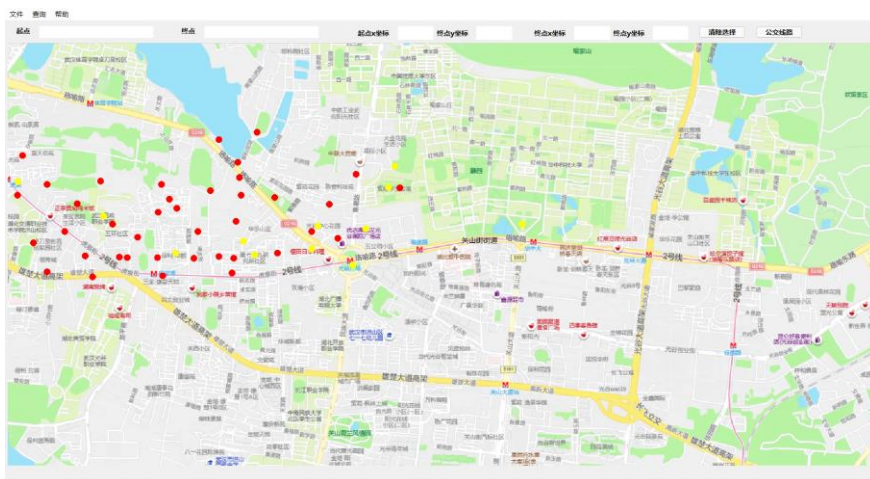


图 5-4-1 加载文件后默认显示图

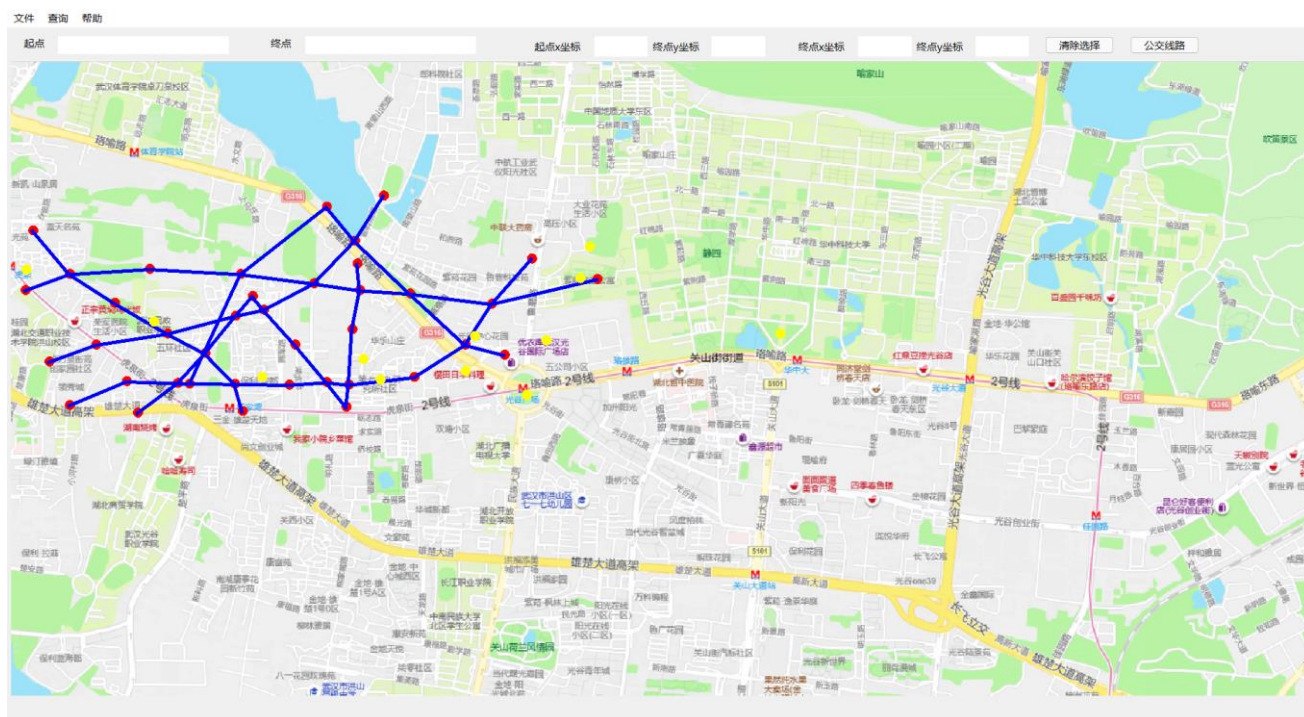


图 5-4-2 加载文件后显示公交线路图

绘制过程中，每次绘制均需要调整坐标系，将原点调整至图片左下角并旋转坐标系到恰当位置，使用到的 Qpainter 成员函数为 translate（调整原点），rotate（旋转坐标系），drawPixmap（绘制 QPixmap 类地图）；然后将公交站点、公交路线、机构位置以不同颜色绘制在地图上；最后使用 addItem 函数为起点、终点、查询子窗口三个 QcomboBox 添加机构名称信息，并默认将其设置为可编辑、无显示内容（index 为-1）。

III涉及到的具体函数分析：

- void MainWindow::paintEvent(QPaintEvent \*)

该函数为 mainwindow 的成员函数，为绘制事件函数，每当调用 update 或 repaint 函


数或者窗口发生变化时都将进行重绘事件。在函数中，首先绘制地图，使用 `QPixmap` 加载地图，并按照适当的大小等比例绘制在窗口中；然后改换坐标系，由于 Qt 默认坐标系以窗口左上角为圆点，右为 x 轴正方向，下为 y 轴正方向，因此需要进行改变原点为地图左下角，旋转坐标系等操作以便于后续选择。接下来，根据为各个需要绘制的量的 flag 来确定是否需要绘制该量，如是否绘制公交站点，是否绘制公交线路，是否绘制机构位置，能否绘制用户选择的点和推荐路线。

### (3) 用户交互选择：

#### I 通过鼠标事件监控用户选择

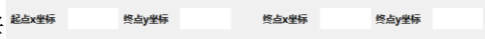
用户选择起点和终点的第一种方式是通过鼠标点击选择。该程序通过使用 `void MainWindow::mousePressEvent(QMouseEvent *event)` 函数实时监控鼠标事件。当文件未加载完成时不进行鼠标监控，以免引发程序错误。当文件加载完成后，即开始进行鼠标监控，如果检测到左键单击 (`Qt::LeftButton`) 事件，则更新起点结构体 (`choosef`) 信息，其中包括被选择起点的坐标值，并实时检查该起点是否在某个公交站点或者机构处 (如果是在某机构处，则更新起点 `QcomboBox` 的显示信息)，以提高决策效率。同样的，如果检测到右键单击 (`Qt::RightButton`) 事件，则更新终点结构体 (`chooset`) 信息并进行相关操作。不论是先选择起点还是终点，一旦起点和终点选择完成后，程序将会自动进入 GIS 类的成员函数 `minitrans` 决策最少转乘路线并进行重绘以呈现给用户。

#### II 通过起点和终点 `QcomboBox` 监控用户选择的机构

在  栏目中，用户可以输入机构名称模糊补全查找机构并选择之，也可以通过下拉列表选择。当选定起点或终点后，程序将实时在地图上显示其位置，当起点和终点均选择完成后，程序会立即进入 `minitrans` 函数决策路线并重绘呈现。

#### III 通过菜单栏项“查询-查询地点”或者快捷键 `ctrl+F` 监控用户选择

当用户打开查询功能后，将会在窗口中心位置弹出查询机构子窗口，该窗口由 `QDialog` 实现，其中包括查询 `QcomboBox`，查询提示和退出按钮。用户可以输入机构名称模糊补全查找机构并选择之，也可以通过下拉列表选择机构，选择完成后，将会在地图中实时突出机构位置，每次进入查询功能可进行多次查询。

每次选择点位时，均会在起点和终点的坐标信息栏  更新坐标信息，方便用户用于修改 `lines`、`stops`、`organization` 文件内容。

#### IV 涉及到的具体函数分析：

- void MainWindow::mousePressEvent(QMouseEvent \*event)

这是 mainwindow 的成员函数，用于监控鼠标事件。首先，如果文件尚未加载完成则不进行鼠标监控退出函数；如果文件已经加载完成，则用户的每一步点击鼠标事件都会被实时监控。如果点击鼠标左键，则将 choosef 的 flag 置为 1，代表起点已经选择，然后获得当前鼠标相对于新的坐标系的位置并存入 choosef 结构体中，同时更新窗口中的坐标信息，为了方便决策，还实时判断用户选择的点是否在某个站点上或者机构处，以上步骤完成后，立即调用重绘事件，将用户选择的起点绘制在地图中 g；点击鼠标右键同上，更新 chooset 中的终点信息。

- void MainWindow::choosebegin(int index)

- void MainWindow::choosefinish(int index)

- void MainWindow::focusorg(int index)

以上三个函数用来监控用户直接通过下拉菜单选择机构位置来确定起点和终点，直接将机构的坐标传入 chooset 和 choosef 结构体中，并进行重绘。

#### (4) 方案的决策与图形化显示：

##### I 进入决策函数 minitran

决策函数有三个参数——choosef,chooset,ans，分别存储起点信息，终点信息，最终路线结果。首先，搜索距离起点和终点的最近站点，如果已经在鼠标监控事件中获得临近站点则无需搜索；然后搜索各得到站点所在的线路，接着按照组合数依次比较各线路之间转乘时的转乘次数，将转乘次数最少的方案存入 ans 的成员 r 并将转乘次数存入成员 n 中，最后回溯查找上车和下车站点并存入 ans 成员 from 和 to 中，决策完毕。

##### II 使用 ans 信息重绘以呈现最少转乘方案

获得 ans 后，通过更新 ans->flag 信息以绘制路径方案。绘制过程为绘制用户选择的起点和终点，绘制上车站点和下车站点，绘制中间转乘站点，绘制线路。该过程使用到的 Qt 类及其成员函数有 QPainter 及其成员函数 save、restore、setPen、setBrush、drawEllipse、drawLine，以及 QLine。

##### III 涉及到的具体函数和数据结构分析：

- void GIS::miniTran(CHOICE \*from,CHOICE\*to,ANS \*ans)

该函数为 GIS 的成员函数，参数为两个 CHOICE 结构体指针（该结构体用来存储用户选择的起终点的相关信息）和指向存储结果信息的 ANS 结构体指针。寻找最少转乘路线的具体方法分为四步。第一步搜索距离起点和终点的最近站点，如果已经提供

临近站点则无需搜索，第二步搜索各得到的站点所在线路，第三步按照组合数依次比较各线路之间转乘时的转乘次数，第四步按照各极小转乘路径的转乘次数升序排序得到最小转乘路径方案并存入 `ans` 数组中。

- `int* GIS::nearstop(int x, int y)`

该函数为 `miniTran` 函数的辅助函数，用来查找距离坐标(x,y)最近的站点并将站点编号存入数组返回。其中使用库函数 `qsort` 进行快速排序（`comp` 函数需要自定义）。

- 该过程中用到的辅助数据结构有如下：

`typedef struct info{//存储点距离某 order 车站的距离 dis`

`double dis;`

`int order;`

`}INFO;`

`typedef struct stop_line{//存储车站 stop 所在的线路信息 line，共处于 nofl 条线路上`

`int stop;`

`int nofl;`

`int* line;`

`}SL;`

`typedef struct choice{//存储用户选择的点的信息，包括点的坐标，点是否在车站处`

`int x;`

`int y;`

`int flag;//标记是否已经选择过`

`int stop;`

`}CHOICE;`

### 三、软件开发

本项目使用的系统为 Windows 11（版本号 22000.276），在 Qt6.2.1 和 MSVC2019 64bit 环境下，使用 Qt Creator 4.4.1 开发并调试运行，使用 `windeployqt.exe` 命令行工具和 Enigma Virtual Box 工具将程序以及依赖的 Qt 库与 MSVC 库打包得到 Release 版本单文件可执行程序。

### 四、软件测试

进入地图后，首先加载地图文件，然后加载站点等其他三个文件，加载完成后如图 5-3-1，默认展示地图，以红色点表示公交站点，以黄色点表示机构位置，并默认不显示公交线路，



可以通过“公交线路”按钮显示或隐藏之，详细界面见 5-4-1 和 5-4-2。

单击鼠标左键选取起点，单击鼠标右键选取终点，当选择完成后会自动规划最少转乘路线并以蓝色线路展示，同时用户的起点和终点以较大蓝色圆点显示，上车、下车和转乘站点以较大黑色圆点显示，如图 5-5 所示：

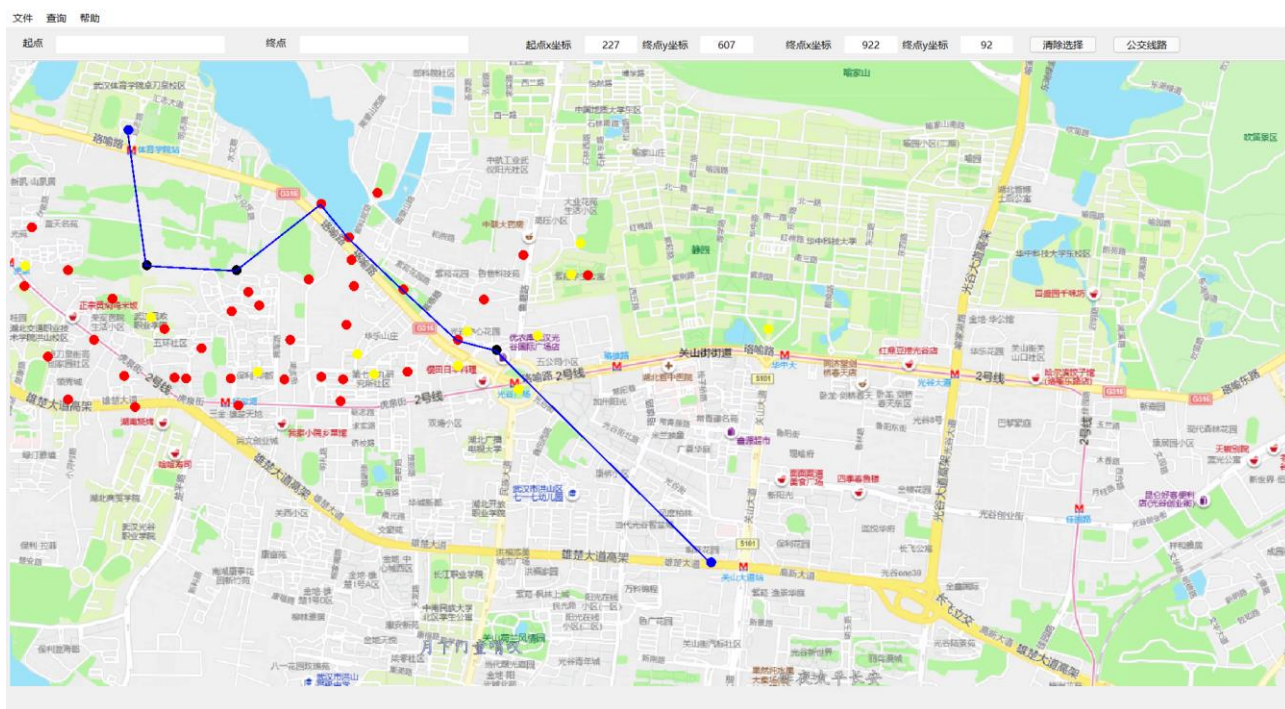


图 5-5 选择起终点后规划路线图

用户可在起点和终点框中下拉菜单选择起点和终点机构，也可以直接模糊输入机构名称选择，选择完成后会给出规划路线，如图 5-6-1 和 5-6-2 所示：

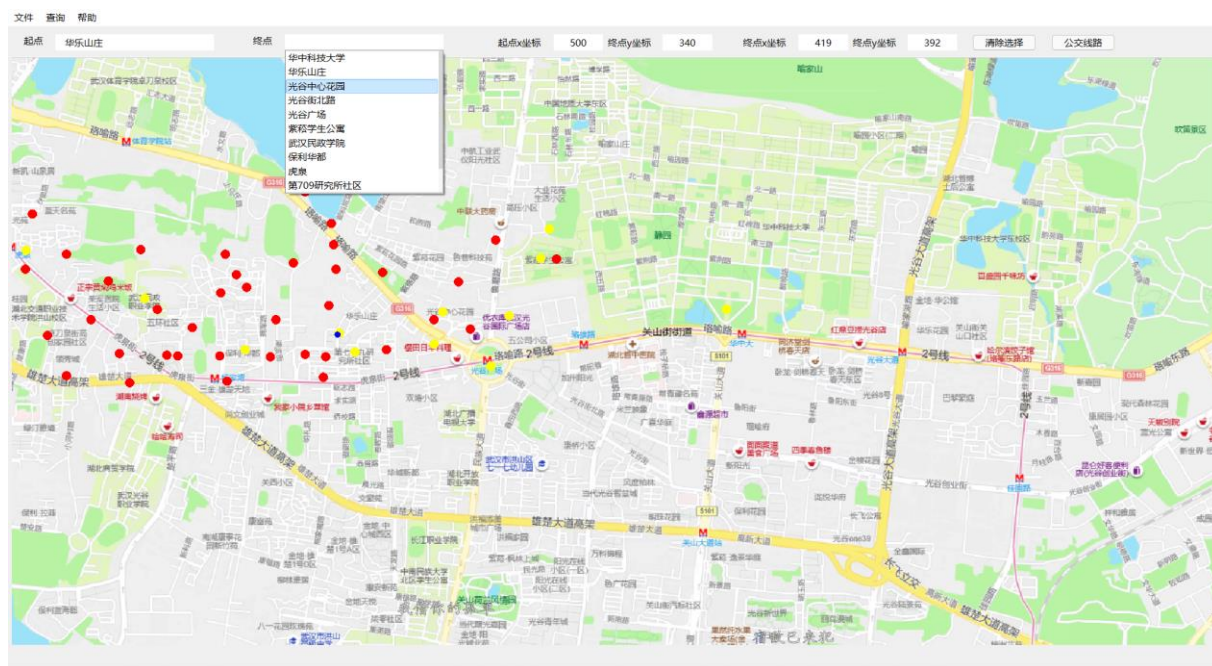


图 5-6-1 通过下拉列表选择起终机构图

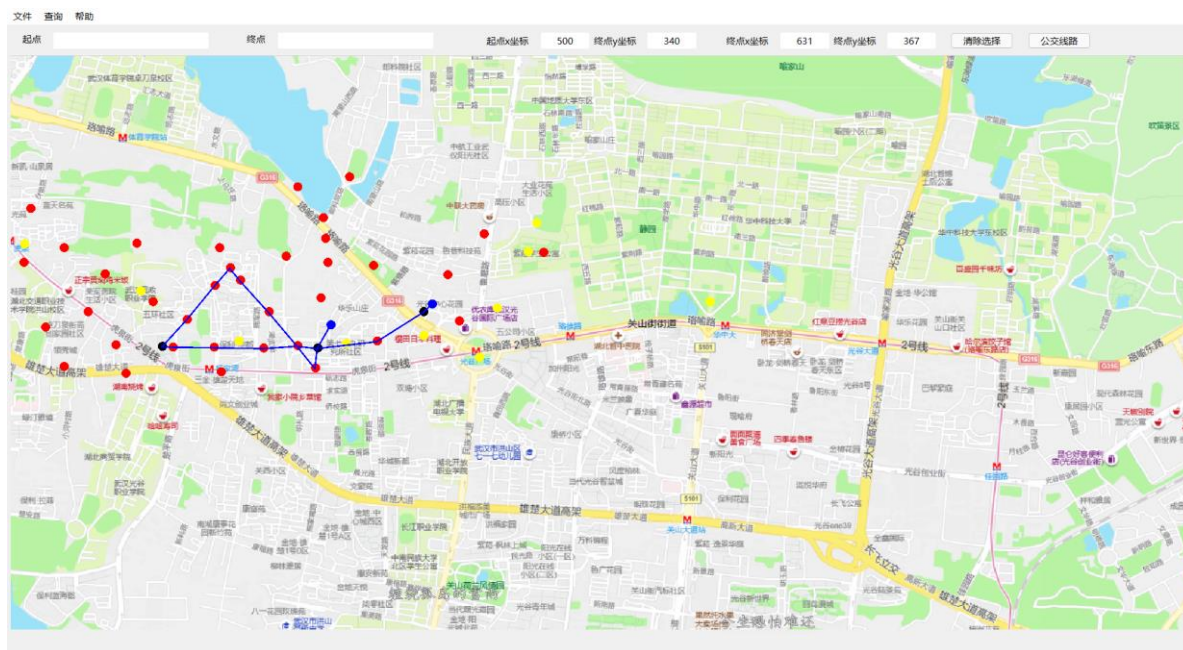


图 5-6-2 通过下拉列表选择起终机构后规划路线图

用户可在查询菜单栏打开查询窗口，查询到某个机构后会在地图中以较大蓝色圆点显示机构位置，如 5-7-1 至 5-7-2 所示

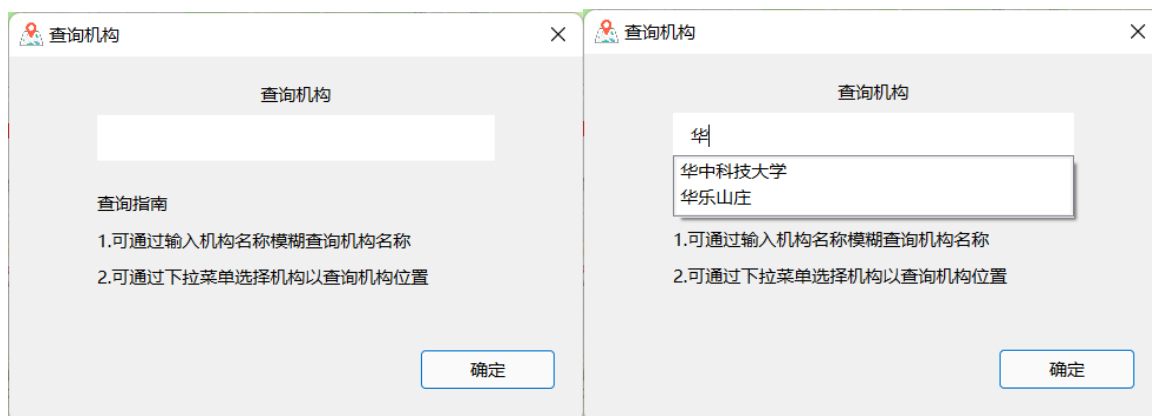


图 5-7-1 查询机构窗口图





图 5-7-2 查询机构显示图

用户可在帮助菜单栏中查看使用帮助，快捷键帮助，版权信息，如图 5-8-1 至 5-8-2 所示

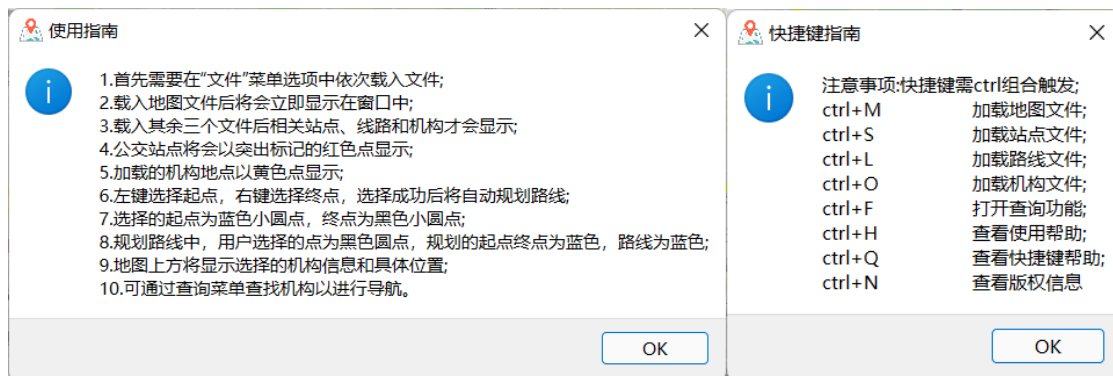


图 5-8-1 使用帮助和快捷键指南图

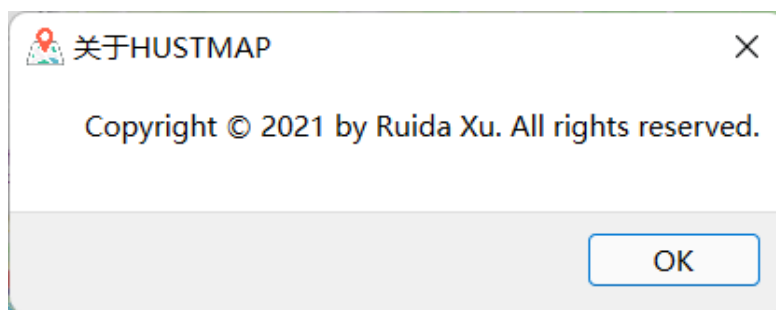


图 5-8-2 版权信息图

## 五、特点与不足

### 1. 技术特点

- (1) 用户可以使用快捷键使用全部功能；
- (2) 用户可以实时查看坐标位置，以便于自行校正数据信息；
- (3) 用户可以选择是否显示公交线路，避免路线杂乱；
- (4) 程序以及菜单栏控件均有对应的图标，美观度得以提升；
- (5) 用户可以通过选择机构来确定终点和起点。
- (6) 将 Qt 依赖库和 MSVC 依赖运行库进行打包，形成单文件可执行 HUSTMAP 程序

### 2. 不足和改进的建议

- (1) 对于一些极端的起点和终点选择，程序可能会出现崩溃闪退的情况；
- (2) QPainter 绘制地图时与实际图片清晰度不匹配，需要改进图片显示方法；
- (3) 突出显示某信息时不能闪烁提示，用户体验可能会受到影响；
- (4) 无法在程序内由用户实时增删改机构信息，后续可以添加该功能。



## 六、过程和体会

### 1. 遇到的主要问题和解决方法

- (1) 不了解 Qt 平台，但查阅 Qt 的 API 文档后，发现，Qt 由 C++ 实现，只不过是许多自定义的类，在熟悉了相关可能会用到的类后，完成项目的难度得以降低；
- (2) 对于存储站点、路线、转乘路径、闭包的数据结构（类）无从下手，在仔细厘清数据之间的关系后，定义了 STOP、LINE、TRAN、ROUTE、NODE、TMAP 等主要类来存储相关数据信息。

### 2. 课程设计的体会

通过这次课程设计，我学会了如何创建并完成 Qt 项目，这也告诉我当遇到新的平台环境时，要学会使用其 API 文档来快速入门。同时，我也对 C++ 类有了更为深入的了解，并逐渐掌握了如何自定义类及其成员函数。

## 七、文件与使用说明

### 1. 文件清单及其功能说明

ex5/project/include 文件夹包括头文件 file.h、mainwindow.h、searchdialog.h, UI 文件 mainwindow.ui 和 searchdialog.ui 及 UI 文件对应头文件；ex5/project/src 文件夹包括实现头文件中函数的对应源文件和 main 函数所在源文件 main.cpp；ex5/project/file 文件夹存放用到的数据文件；另外 ex5/project 中包含可直接运行的单文件 HUSTMAP.exe（该文件已经打包 Qt 和 MSVC 运行库，可直接在裸机上运行）。

### 2. 用户使用说明书

打开 ex5/project/文件夹下 HUSTMAP.exe 文件运行，先在文件菜单栏中依次加载 file 目录下文件（也可以使用快捷键），然后进入用户操作环节，使用完毕后即可在右上角关闭退出。