
哈爾濱工業大學（深圳）
Harbin Institute of Technology Shenzhen

# 《创新训练课 B》结题报告

学院：机电工程与自动化学院

题目：地铁购票系统

（Subway ticketing system）

学生一：psp

上交日期：2023.12.28

# 修订历史记录

| 日期 | 版本 | 说明 | 作者 |
|---|---|---|---|
| 23.7.30 | V1.0 | 搭建完程序框架，学习相关算法、实现基本的功能 | psp |
| 21.8.5 | V1.5 | 使用 Easyx 图形化页面完善程序 | psp |
| 21.8.20 | V2.0 | 完善程序的所有功能，优化了部分执行逻辑 | psp |

**目录**

# 一、引言

## 1.1 编写目的

《高级语言程序设计》课程设计是高级语言程序设计课程学习完毕后进行的一项重要实践性任务，旨在通过全面综合的练习来提升学生的编程能力。该课程要求学生熟练掌握 C 语言的基本知识和技能，并基本掌握面向过程程序设计的思想和方法。学生需要运用所学的基本知识和技能，解决一些简单的面向过程程序设计问题，从而提高他们动手编程解决实际问题的能力。通过这个课程设计，学生将能够更深入地理解 C 语言的应用和实践，培养他们的逻辑思维和问题解决能力。课程设计还为学生提供了一个实践的机会与平台，让他们能够将所学的理论知识应用到实际项目中，提升他们的实践能力和团队合作能力。通过完成 C 语言程序设计课程设计，学生将建立起坚实的编程基础，并为进一步学习和应用计算机编程打下坚实的基础。

## 1.2 背景

随着城市人口的不断增长和交通需求的日益增加，地铁系统作为一种高效、快捷的公共交通工具在现代城市中扮演着重要的角色。然而，传统的人工售票方式在高峰期往往导致长队和拥堵，给乘客带来不便和时间浪费。

为了提升地铁乘客的出行体验，简化购票流程并提高运营效率，设计一个地铁购票系统的程序成为迫切的需求。这个程序旨在利用计算机技术，实现地铁车站的自助售票服务。

这个地铁购票系统的设计旨在提供便捷、高效的购票体验。乘客可以更快捷地购买车票，并且得知起始站点到目的站点间价格最少的路径以及最少的价格。

## 1.3 定义

在地铁购票系统程序中，我们设计了一个自助售票系统，旨在提供便捷、高效的地铁票务服务。该系统通过计算机技术和自动化手段，实现以下功能：

车票购买：乘客可以通过自助售票机选择起始站点和目的站点，并选择合适的票种进行购买。系统将根据乘客选择的行程段和票种计算票价，并提供支付方式供乘客选择。

路线规划：系统将根据乘客选择的起始站点和目的站点，计算出最佳的乘车路线，以方便乘客选择最合适的路线。

用户界面：系统提供简单易懂的用户界面，使乘客能够轻松操作自助售票机。界面应具有友好的交互设计，提供明确的指导和反馈，以确保乘客能够顺利完成购票流程。

## 1.4 参考资料

easyX 使用文档：

https://docs.easyx.cn/zh-cn/intro

弗洛伊德（Floyd）算法求图的最短路径：

https://blog.csdn.net/jeffleo/article/details/53349825

# 二、任务概述

该项目实现了模拟地铁自动售票系统，基础功能：

（1）显示欢迎界面，作者信息和版权信息。

（2）进入系统主菜单，提供购票选项、地图查询选项、退出系统三个选项。

（3）系统说明界面详细的介绍了购票流程，并且附有用户须知。

（4）用户选择开始购票，进入始发站选择界面，或者由此返回主界面。

（5）用户选择好了始发站后进入终点站的选择，或者由此返回主界面。

（6）根据系统提示然后进入票数的选择，或者由此返回主界面。

（7）根据系统提示进入投币找币流程，或者由此返回主界面。

（8）购票成功。

# 三、需求分析

## 3.1 需求分析

在设计地铁购票系统程序时，我们需要满足以下需求：

车票查询和购买：用户应该能够查询地铁线路、车站信息以及车票价格。一旦用户选择起始站点和目的站点，系统应该能够计算并显示最少票价的乘车路线。用户可以根据需要选择合适的车票进行购买。

用户界面设计：系统应该具有直观、友好的用户界面，提供明确的操作指导和反馈信息，以确保用户能够轻松地完成购票流程。

错误处理和异常情况处理：系统应该能够处理用户输入错误、网络异常等情况，并给出相应的提示和解决方案，以提供良好的用户体验。

## 3.2 运行环境

操作系统：windows 11

UI：easyx

# 四、功能及操作介绍

## 4.1 操作

根据可视化页面的提示选择选项或输入数据，便可得到相应的结果。

# 五、系统设计

## 5.1 总体架构设计

在地铁购票系统的总体架构设计中，我们采用了以下的设计思路和模块分析：

类层次设计、系统运行层次设计、软件结构（流程图）设计。

通过以上的总体架构设计，我们将地铁购票系统划分为不同的模块，每个模块负责特定的功能，使程序的结构清晰、模块化，便于后续的开发、调试和维护。

## 5.2 模块分析与设计

本程序通过 C++语言编写，使用的 UI 引擎为 easyx：

### 5.2.1 类层次：

程序中类的层次如下：

一、站点（Station）：是地图中的“点”，包含站点的名称和唯一标识符。

类的声明源代码：

```
    // 站点
    typedef struct Station_t {
        string name;
        char id = 0;
    } Station;
```

二、路线（Line）：表示地图中两个站点的连接，是地图中的“边”，包含起始站点、目标站点和票价。其中 price 将会作为之后计算两个站点件最少价格的“权”。

类的声明源代码：

```
    // 路线（连接两个站点）
    typedef struct Line_t {
        Station station_1;
        Station station_2;
        unsigned int price = 0;  // 两站间的价格
    } Line;
```

三、地图（Map）：包含了整个地图的所有信息，包括有哪些站点、站点之间如何连接、站点之间的路线的距离之类的信息。地图类具有计算最短路径和最小金额的方法，并提供购票相关的逻辑。

类的声明源代码：

```
    // 地图
    class Map {
       private:
        Station station[NUM_OF_STATION];
        Line line[NUM_OF_LINE];
        int graph[NUM_OF_STATION][NUM_OF_STATION];  // 邻接矩阵存图
        int path_matrix[20][20];                    // 记录对应点的最小路径的前驱点，例如 path_matrix[1,3] = 2 说明顶点 1 到顶点 3 的最小路径要经过 2
        int short_path[20][20];                     // 记录顶点间的最小路径值

        // 计算最短路径，并存在矩阵中
        void short_path_Floyd();

```

```
    // 输入起始站点和目标站点，返回最小金额
    int GetCost(const int station_id_1, const int station_id_2) const;

public:
    explicit Map();

    // 购票相关逻辑-图形化版
    void Buy_Tickets_easyx() const;
};
```

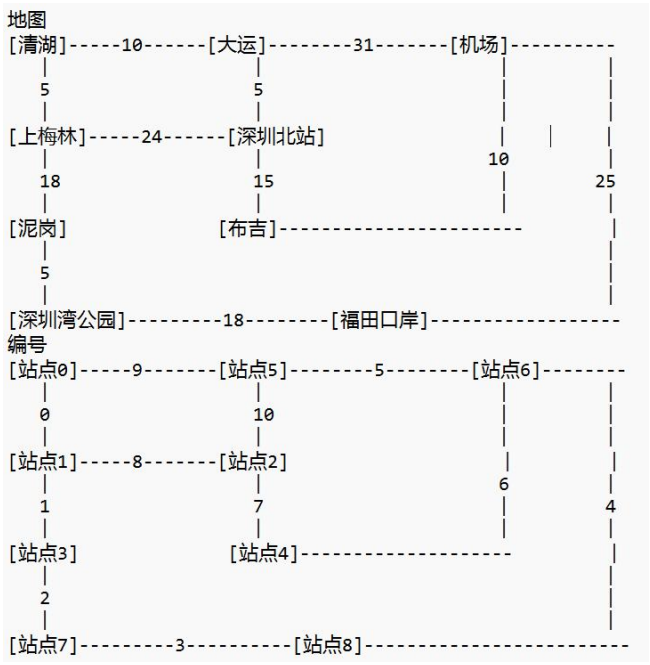
其中类方法的实现代码如下：

构造方法：进行了数据的初始化。

```
Map::Map(){
    // 初始化地图数据
    // 创建站点
    station[0].name = "清湖";
    station[0].id = 0;
    station[1].name = "上梅林";
    station[1].id = 1;
    station[2].name = "深圳北站";
    station[2].id = 2;
    station[3].name = "泥岗";
    station[3].id = 3;
    station[4].name = "布吉";
    station[4].id = 4;
    station[5].name = "大运";
    station[5].id = 5;
    station[6].name = "机场";
    station[6].id = 6;
    station[7].name = "深圳湾公园";
    station[7].id = 7;
    station[8].name = "福田口岸";
    station[8].id = 8;
    // 创建线路
    line[0].station_1 = station[0];
    line[0].station_2 = station[1];
    line[0].price = 5;
    line[1].station_1 = station[1];
    line[1].station_2 = station[3];
    line[1].price = 18;
    line[2].station_1 = station[3];
    line[2].station_2 = station[7];
    line[2].price = 5;
    line[3].station_1 = station[7];
    line[3].station_2 = station[8];
    line[3].price = 18;
    line[4].station_1 = station[8];
    line[4].station_2 = station[6];
    line[4].price = 25;
    line[5].station_1 = station[5];
    line[5].station_2 = station[6];
    line[5].price = 31;
    line[6].station_1 = station[4];
    line[6].station_2 = station[6];
    line[6].price = 10;
    line[7].station_1 = station[2];
```

```
45.         line[7].station_2 = station[4];
46.         line[7].price = 15;
47.         line[8].station_1 = station[1];
48.         line[8].station_2 = station[2];
49.         line[8].price = 24;
50.         line[9].station_1 = station[0];
51.         line[9].station_2 = station[5];
52.         line[9].price = 10;
53.         line[10].station_1 = station[2];
54.         line[10].station_2 = station[5];
55.         line[10].price = 5;
56.         // 邻接矩阵初始化
57.         for (int i = 0; i < NUM_OF_STATION; i++)
58.             for (int j = 0; j < NUM_OF_STATION; j++)
59.                 if (i == j)
60.                     graph[i][j] = 0;
61.                 else
62.                     graph[i][j] = INF;
63.
64.         for (int i = 0; i < NUM_OF_LINE; i++) {  // 将距离存入邻接矩阵
65.             graph[line[i].station_1.id][line[i].station_2.id] = line[i].price;
66.             graph[line[i].station_2.id][line[i].station_1.id] = line[i].price;
67.         }
68.         short_path_Floyd();
69.     }
```

以下是程序中的线路图的直观表示，第一个图包含站点名与路线的价格，第二个图包含站点编号与路线的编号




Map::short_path_Floyd 方法：计算所有的两个站点之间的最小金额并存进矩阵里，方便后续输入任意两个站点时输出最小金额以及最优路径：

```
1.     // 计算最少金额，并存在矩阵中
2.     void Map::short_path_Floyd() {
3.         int v, w, k;
4.         // 初始化 Floyd 算法的两个矩阵
5.         for (v = 0; v < NUM_OF_STATION; v++) {
```

```
            for (w = 0; w < NUM_OF_STATION; w++) {
                short_path[v][w] = graph[v][w];
                path_matrix[v][w] = w;
            }
        }
        // 这里是弗洛伊德算法的核心部分
        // k 为中间点
        for (k = 0; k < NUM_OF_STATION; k++) {
            // v 为起点
            for (v = 0; v < NUM_OF_STATION; v++) {
                // w 为终点
                for (w = 0; w < NUM_OF_STATION; w++) {
                    if (short_path[v][w] > (short_path[v][k] + short_path[k][w])) {
                        short_path[v][w] = short_path[v][k] + short_path[k][w];  // 更新最小金额
                        path_matrix[v][w] = path_matrix[v][k];                  // 更新最小金额中间顶点
                    }
                }
            }
        }
        return;
    }
```

这里使用了弗洛伊德算法（Floyd）来寻找两个站点的最短路径。

弗洛伊德算法基本思路：

弗洛伊德算法定义了两个二维矩阵：

1、矩阵 D（在代码中是 short_path）记录顶点间的最小路径

（例如 D[0][3]= 10，说明顶点 0 到 3 的最短路径为 10）

2、矩阵 P（在代码中是 path_matrix）记录顶点间最小路径中的中转点

（例如 P[0][3]= 1 说明，0 到 3 的最短路径轨迹为：0 -> 1 -> 3）

通过 3 重循环，k 为中转点，v 为起点，w 为终点，循环比较 D[v][w] 和 D[v][k] + D[k][w] 最小值，如果 D[v][k] + D[k][w] 为更小值，则把 D[v][k] + D[k][w] 覆盖保存在 D[v][w]中。

弗洛伊德算法作为求最短路径的经典算法，其算法实现具有不错的可读性和性能。

Map::GetCost 方法输入两个站点，返回最小金额：

```
    // 输入起始站点和目标站点，返回最小金额
    int Map::GetCost(const int station_id_1, const int station_id_2) const {
        int v = station_id_1;
        int w = station_id_2;
        cout << endl
             << station[v].name << " -> " << station[w].name << " 的单张票最小金额为：" << short_path[v][w] << endl;
        int k = path_matrix[v][w];
        cout << "路径为：" << station[v].name;
        while (k != w) {
            cout << " -> " << station[k].name;
            k = path_matrix[k][w];
        }
        cout << " -> " << station[w].name << endl;
        return short_path[v][w];
    }
```

Map::Buy_Tickets_easyx 方法：购票的相关逻辑

```
1.  // 购票相关逻辑-图形化版
2.  void Map::Buy_Tickets_easyx() const {
3.      cleardevice();  // 刷新
4.      IMAGE background;
5.      loadimage(&background, "./src/picture/roadmap.jpg", 1080, 720);  // 从图片文件获取图像
6.      putimage(0, 0, &background);                                      // 绘制图像到屏幕，图片左上角坐标为(0,0)
7.      bool is_start = false, is_end = false, is_ticket = false;
8.      char start_station_input[10] = {};
9.      char end_station_input[10] = {};
10.     char numofTicket_input[10] = {};
11.     int numofTicket = 0;  // 购买的票数
12.     Station station_1, station_2;
13.
14.     TCHAR button1[50] = "请输入起点";
15.     TCHAR button2[50] = "请输入终点";
16.     TCHAR button3[50] = "请输入票数";
17.     TCHAR button4[50] = "确认";
18.     TCHAR button5[50] = "退出购票";
19.     button(391, 200, 250, 50, button1);
20.     button(391, 300, 250, 50, button2);
21.     button(391, 400, 250, 50, button3);
22.     button(391, 500, 250, 50, button4);
23.     button(391, 600, 250, 50, button5);
24.     while (true) {
25.         FlushBatchDraw();
26.         ExMessage msg;
27.         if (peekmessage(&msg, EM_MOUSE)) {
28.             while (peekmessage(&msg, EM_MOUSE))
29.                 ;
30.             if (msg.lbutton)  // 如果按下了左键
31.             {
32.                 bool is_OK = true;
33.                 if (msg.x >= 391 && msg.x <= 391 + 250 && msg.y >= 200 && msg.y <= 200 + 50 && !is_start) {
34.                     printLog("输入起点");
35.                     while (!is_start && is_OK) {
36.                         is_OK = InputBox(start_station_input, 20, "请输入起点", "您正在输入起点站", 0, 0, 0, false);  // 按“确定”，返回 true；按“取消”，返回 false
37.                         if (is_OK) {
38.                             bool is_station = 0;
39.                             for (int i = 0; i < NUM_OF_STATION; ++i) {
40.                                 if (station[i].name == start_station_input) {
41.                                     is_station = 1;
42.                                     station_1 = station[i];
43.                                     break;
44.                                 }
45.                             }
46.                             if (is_station) {
47.                                 printLog("输入成功");
48.                                 outtextxy(650, 200, "起点：");
49.                                 outtextxy(730, 200, start_station_input);
50.                                 is_start = true;
51.                             } else {
52.                                 printLog("您的输入有误，请重新输入");
```

```
                            }
                        }
                    }
                } else if (msg.x >= 391 && msg.x <= 391 + 250 && msg.y >= 300 && msg.y <= 300 + 50 && !is_end) {
                    cout << "输入终点" << endl;
                    while (!is_end && is_OK) {
                        bool is_OK = InputBox(end_station_input, 20, "请输入终点", "您正在输入起点站", 0, 0, 0, false);  // 按“确定”，返回 true；按“取消”，返回 false
                        if (is_OK) {
                            bool is_station = 0;
                            for (int i = 0; i < NUM_OF_STATION; ++i) {
                                if (station[i].name == end_station_input && end_station_input != station_1.name) {
                                    is_station = 1;
                                    station_2 = station[i];
                                    break;
                                }
                            }
                            if (is_station) {
                                printLog("输入成功");
                                outtextxy(650, 300, "终点：");
                                outtextxy(730, 300, end_station_input);
                                is_end = true;
                            } else {
                                printLog("您的输入有误，请重新输入");
                                outtextxy(650, 300, "终点：");
                                outtextxy(730, 300, "未找到此站点，请重新输入");
                            }
                        }
                    }
                } else if (msg.x >= 391 && msg.x <= 391 + 250 && msg.y >= 400 && msg.y <= 400 + 50 && !is_ticket) {
                    cout << "输入票数" << endl;
                    InputBox(numofTicket_input, 20, "请输入票数");
                    numofTicket = atoi(numofTicket_input);
                    outtextxy(650, 400, "票数：");
                    outtextxy(730, 400, numofTicket_input);
                    is_ticket = true;
                } else if (msg.x >= 391 && msg.x <= 391 + 250 && msg.y >= 500 && msg.y <= 500 + 50) {
                    if (is_start && is_end && is_ticket) {
                        cout << "确认" << endl;
                        int total_cost = numofTicket * GetCost(station_1.id, station_2.id);
                        cout << "所需要的总金额为：" << total_cost << endl;
                        outtextxy(650, 500, "总价：");
                        char cost_str[10];
                        sprintf_s(cost_str, "%d", total_cost);
                        outtextxy(730, 500, cost_str);
                    } else {
                        cout << "未完成输入就确认" << endl;
                        MessageBox(NULL, "您未完成输入", "Wrong", MB_OK);
                    }
                } else if (msg.x >= 391 && msg.x <= 391 + 250 && msg.y >= 600 && msg.y <= 600 + 50) {
                    return;
                }
            }
        }
    }
}
```

### 5.2.2 程序运行层次

程序在运行后即进入 main 函数运行，main 函数是程序的入口函数，负责创建绘图窗口、处理用户输入和执行相应的逻辑。在 main 函数中完成前述的循环等待鼠标点击、判断鼠标点击的地方并执行相应逻辑的功能。

main 函数具体提供了如下功能：

1、用户界面（UI）：使用 EasyX 图形库创建用户界面，包括欢迎界面、主菜单、购票界面和线路查询界面等。

2、用户选择（Choice）：根据用户的鼠标点击位置，判断用户选择的操作，如购票、线路查询或退出系统。

3、地铁购票系统（Map）：负责处理用户购票逻辑，包括选择起始站点、目标站点、购票张数和支付等。还提供了购票结果的展示和最短路径查询的功能。

main 函数源代码：

```
1.    int main(int argc, char* argv[]) {
2.        initgraph(1080, 720, EX_DBLCLKS | EX_NOCLOSE | EX_NOMINIMIZE | EX_SHOWCONSOLE);  /
    / 创建绘图窗口
3.        SetWindowText(GetHWnd(), "PSP and LWT's work");                                   /
    / 设置窗口标题
4.
5.        Map map;  // 创建地铁线路图
6.        Choice choice = undefined_e, last_choice = undefined_e;
7.        char result[10] = {};
8.
9.        settextstyle(50, 0, "Consolas");
10.       settextcolor(BLACK);
11.
12.       IMAGE background、menu;                                          // 定义图片
13.       loadimage(&background, "./src/picture/menu.jpg", 1080, 720);  // 从图片文件获取图像
14.       loadimage(&menu, "./src/picture/roadmap.jpg", 1080, 720);     // 从图片文件获取图像
15.
16.       BeginBatchDraw();  // 执行后，任何绘图操作都将暂时不输出到绘图窗口上，直到执
    行 FlushBatchDraw 或 EndBatchDraw 才将之前的绘图输出，防止闪屏
17.       while (true) {
18.           FlushBatchDraw();
19.           if (choice == undefined_e) {
20.               putimage(0, 0, &background);  // 绘制图像到屏幕，图片左上角坐标为(0,0)
21.               TCHAR button1[50] = "1.购票";
22.               TCHAR button2[50] = "2.线路查询";
23.               TCHAR button3[50] = "3.退出系统";
24.               button(21, 530, 200, 50, button1);
25.               button(391, 530, 200, 50, button2);
26.               button(691, 530, 200, 50, button3);
27.               ExMessage msg;
28.               if (peekmessage(&msg, EM_MOUSE))  // 用于获取一个消息，并立即返回。
29.               {
30.                   while (peekmessage(&msg, EM_MOUSE))
31.                       ;
32.                   if (msg.lbutton)  // 如果按下了左键
```

```
                {
                    if (msg.x >= 21 && msg.x <= 21 + 200 && msg.y >= 530 && msg.y <= 5
30 + 50) {
                        choice = buy_tickets_e;
                        if (choice != last_choice) printLog("choice = buy_tickets_e");
                    } else if (msg.x >= 391 && msg.x <= 391 + 200 && msg.y >= 530 && m
sg.y <= 530 + 50) {
                        choice = show_map_e;
                        if (choice != last_choice) printLog("choice = show_map_e");
                    } else if (msg.x >= 691 && msg.x <= 691 + 200 && msg.y >= 530 && m
sg.y <= 530 + 50) {
                        choice = exit_e;
                        if (choice != last_choice) printLog("choice = exit_e");
                    } else {
                        choice = undefined_e;
                        if (choice != last_choice) printLog("choice = undefined_e");
                    }
                }
            }
        }

        if (choice == buy_tickets_e) {
            map.Buy_Tickets_easyx();
            choice = undefined_e;
        }

        else if (choice == show_map_e) {  // 显示路线图
            cleardevice();                // 刷新
            putimage(0, 0, &menu);        // 绘制图像到屏幕，图片左上角坐标为(0,0)
            TCHAR button4[50] = "1.返回主页";
            button(391, 530, 200, 50, button4);
            ExMessage msg;
            if (peekmessage(&msg, EM_MOUSE))  // 用于获取一个消息，并立即返回。
            {
                while (peekmessage(&msg, EM_MOUSE))
                    ;
                if (msg.lbutton)  // 如果按下了左键
                {
                    if (msg.x >= 391 && msg.x <= 391 + 200 && msg.y >= 530 && msg.y <=
 530 + 50) {
                        choice = undefined_e;
                    }
                }
            }
        } else if (choice == exit_e)  // 退出
        {
            return 0;
        }
        last_choice = choice;
    }

    EndBatchDraw();
    closegraph();  // 关闭绘图窗口
    return 0;
}
```

### 5.3 软件结构（流程图）

**一、程序运行层次：**

这张流程图呈现展现给用户的程序的运行逻辑，包括打开程序之后创建绘图窗口、显示欢迎界面和处理用户选择等。

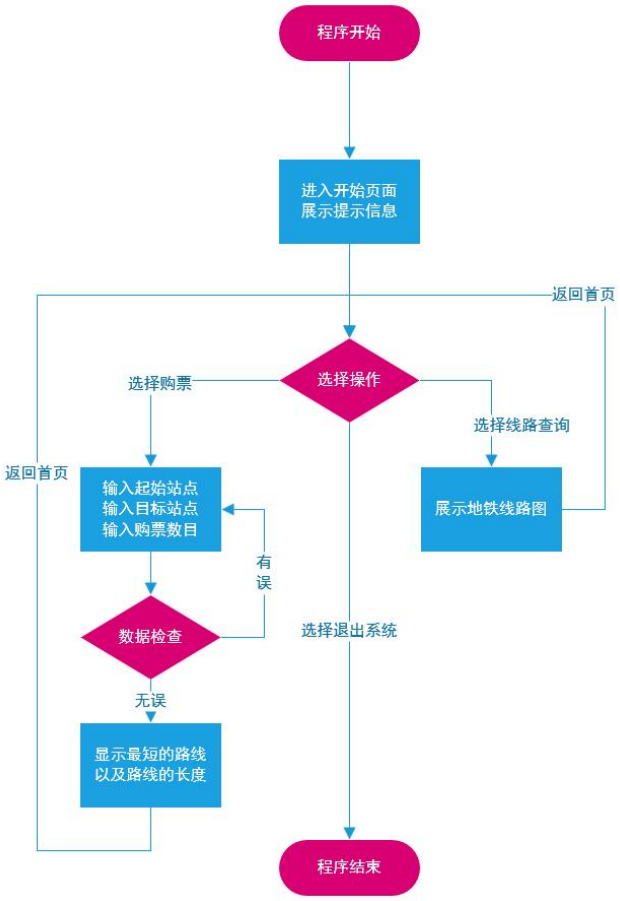



图 1 展现给用户的程序的运行逻辑

**二、程序循环过程层次：**

这张流程图呈现程序内部的执行逻辑以及循环过程。

打开程序后，程序将进行循环，等待用户鼠标点击，如果有鼠标点击，则扑捉并分析点击位置，从而执行相应的逻辑。

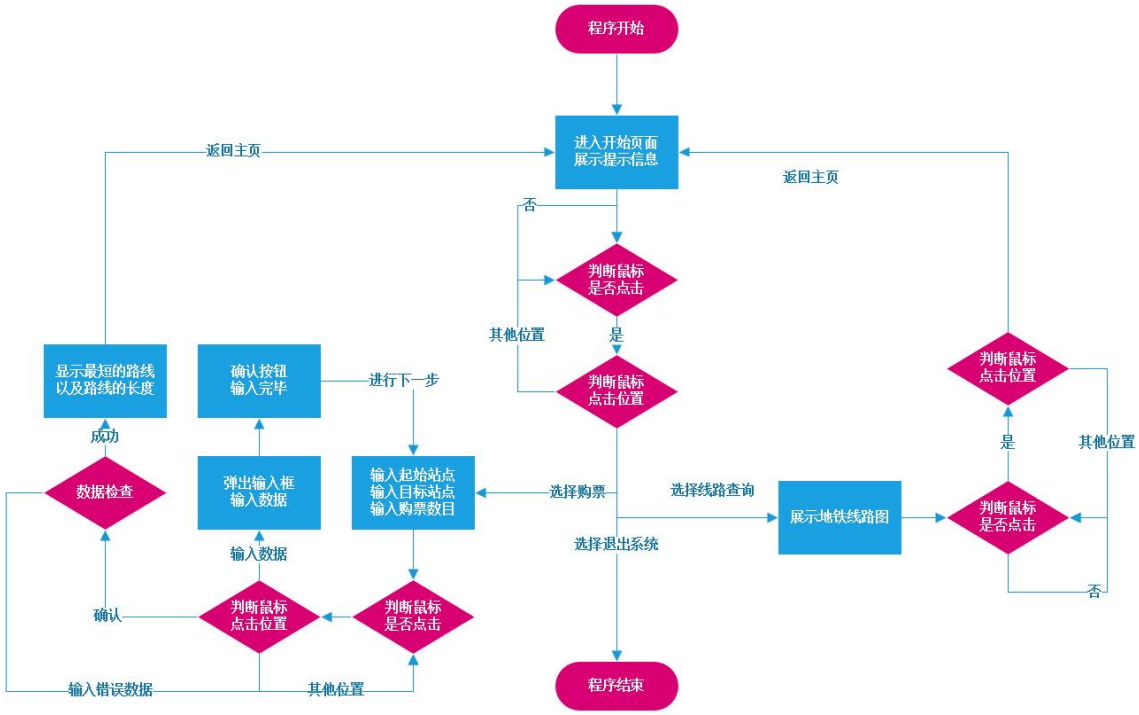



图 2 程序内部的执行逻辑以及循环过程

# 六、调试与测试

## 6.1 调试过程

调试过程是一个重要的步骤，用于确保程序的正确性和稳定性。在进行功能性更新或新增功能后，以下是我们开发这个程序的调试过程：

设计测试用例：根据新增功能或功能性更新的需求，设计一组测试用例，覆盖各种可能的输入情况和边界条件。测试用例包括了正常情况和异常情况，以验证程序在各种情况下的行为是否符合预期。

执行测试用例：按照设计的测试用例，逐个执行测试用例，并记录每个测试用例的输入、预期输出和实际输出。

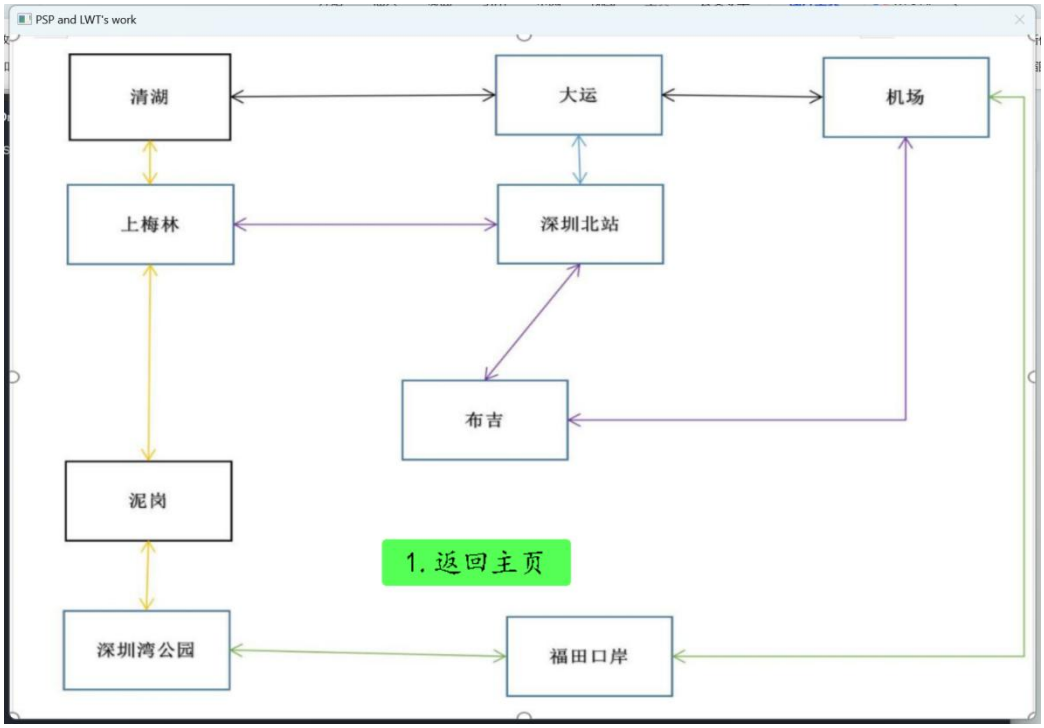
调试错误和异常：如果在执行测试用例的过程中发现错误或异常情况，我们进行调试。首先，根据错误信息或异常提示，定位错误发生的位置和原因。然后，我们根据问题的具体情况，分析错误的原因并修复代码。

验证修复：在修复错误后，我们重新执行相关的测试用例，验证修复是否成功。确保修复后的程序在相同的输入条件下产生预期的输出结果。

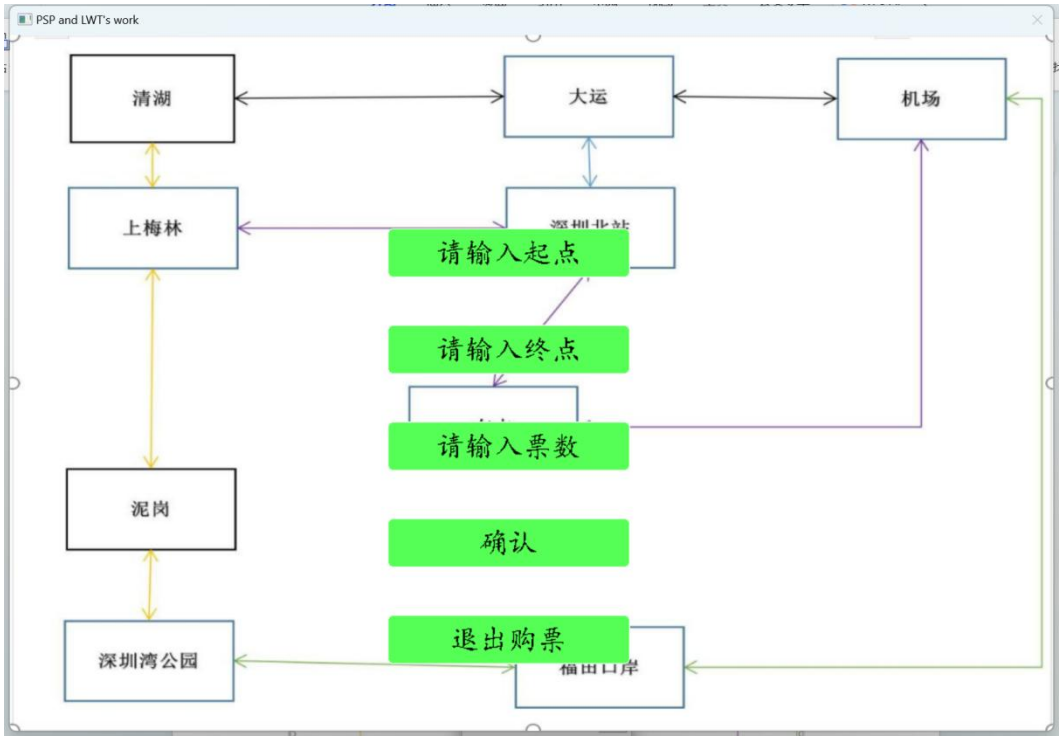
我们通过重复执行以上步骤，直到程序达到预期的功能。

## 6.2 测试结果（包括界面，数据，结果）

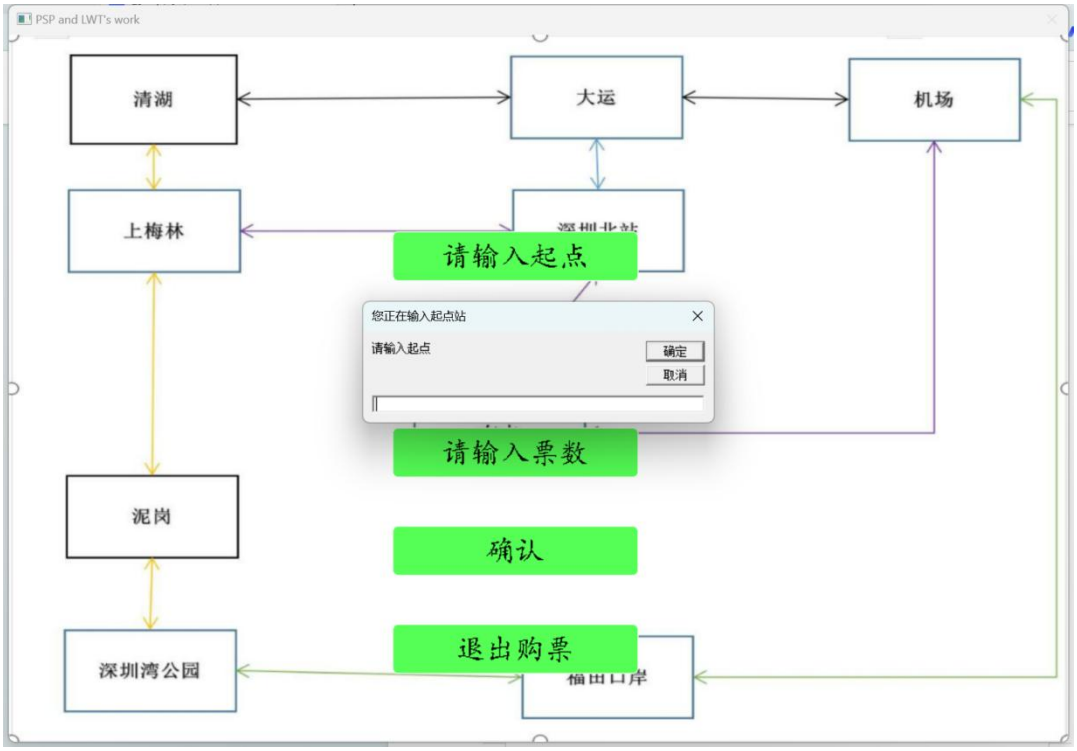
线路查询界面：


PSP and LWT's work
清湖
大运
机场
上梅林
深圳北站
布吉
泥岗
1. 返回主页
深圳湾公园
福田口岸

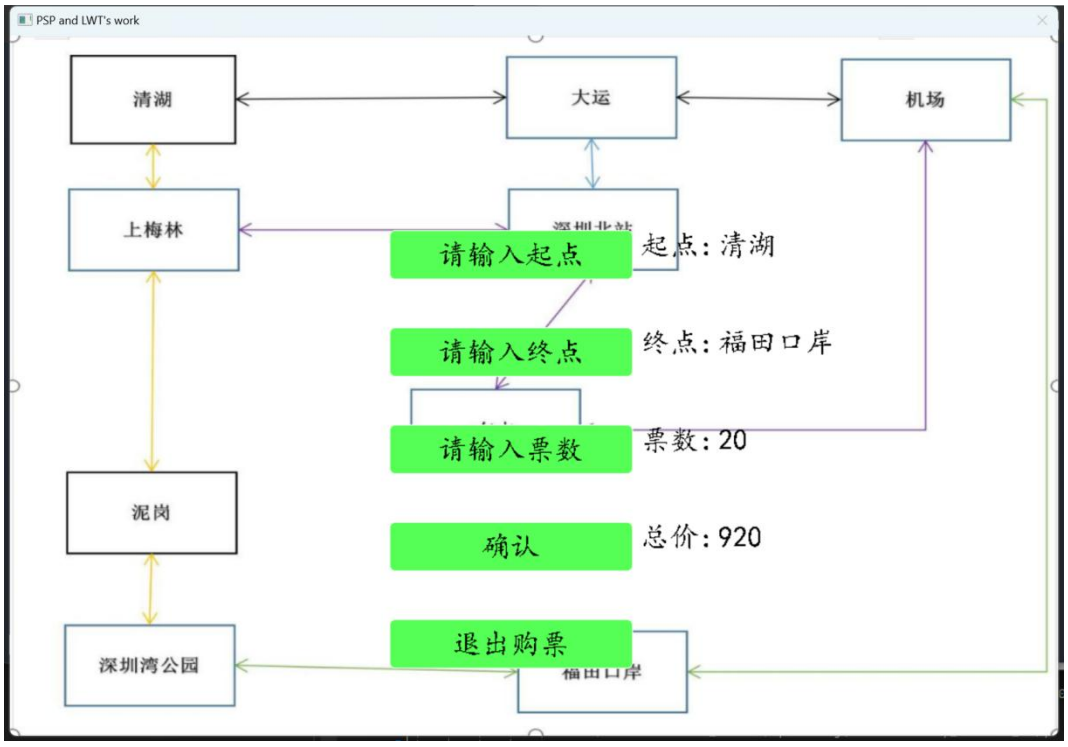
购票界面：


PSP and LWT's work
清湖
大运
机场
上梅林
请输入起点
请输入终点
请输入票数
泥岗
确认
退出购票
深圳湾公园

输入数据：




输入起始站、目标站以及购票张数后返回购票结果：




在控制台会打印对应的数据：

```
清湖 -> 福田口岸 的单张票最小金额为： 46
路径为： 清湖 -> 上梅林 -> 泥岗 -> 深圳湾公园 -> 福田口岸
所需要的总金额为： 920
```

## 七、编程中遇到的问题与解决

问题一：

问题：程序在切换页面的时候会发生闪烁，影响使用体验

解决：Easyx 引擎在不同页面间切换时容易造成画面闪烁。使用 Easyx 提供的缓冲区写入刷出函数使界面中的元素全部同步更新即可解决这个问题。即使用如下函数：

BeginBatchDraw();

这个函数执行后，任何绘图操作都将暂时不输出到绘图窗口上，直到执行 FlushBatchDraw() 或 EndBatchDraw() 函数才将之前的绘图输出，防止闪屏。

问题二：

问题：peekmessage() 是 EasyX 图形库中的一个函数，用于检查程序的消息队列中是否存在特定类型的消息（在这个程序中是鼠标的点击），并且立即返回，不会等待。但是一次鼠标点击产生的消息在一次 while 循环中没有被完全处理，导致在下一次 while 循环时，peekmessage()函数仍然返回这个未处理的鼠标点击消息。

解决：在每次处理完鼠标点击事件后添加如下代码，持续调用 peekmessage()函数，直到消息队列中没有鼠标点击消息为止，这样就可以确保在下一次 while 循环时，peekmessage()函数不会返回未处理的鼠标点击消息。

```
1.      while (peekmessage(&msg, EM_MOUSE));
```

## 八、分析总结与心得体会

通过地铁购票系统项目的设计和开发，我们在 C++编程方面获得了很多知识和实践经验。以下是我们在这个项目中的分析总结和心得体会：

熟悉 C++语言特性：通过这个项目，我们更加熟悉了 C++语言的特性，包括面向对象编程、类和对象的使用等。这些概念在地铁购票系统的设计中得到了广泛应用，帮助我们更好地组织和管理代码。

理解软件开发流程：参与地铁购票系统的开发过程，让我们更加了解了软件开发的流程。从需求分析、设计到编码和测试，每个阶段都需要仔细思考和规划。这种系统化的开发流程有助于提高代码的质量和可维护性。

学会使用图形库：在地铁购票系统中，我们学会了使用 easyx 图形库来创建用户界面。通过绘制按钮、文本框和图形等元素，我能够设计出直观美观的界面，提升用户体验。

锻炼问题解决能力：在项目开发过程中，我们遇到了各种问题和挑战。有时是逻辑错误，有时是界面显示问题，还有时是性能优化需求。通过分析问题、查找资料和与同学交流，我逐渐培养了解决问题的能力，提升了自己的编程技巧。

总的来说，参与地铁购票系统项目的设计和开发，让我们在 C++编程方面得到了很大的提升。除了技术上的收获，我们还学会了团队合作、解决问题和沟通协作的重要性。这些经验将对我们未来的学习和职业发展有很大的帮助。