

# 设计说明书

## 一、 类的设计说明

本程序中设计了六个类，包括公交车类，位置坐标类，公交车站类，线路类，时间类，用户信息类，以下是各个类的具体说明：

### 1. 公交车类：Bus

该类是用于记录公交车的基本信息和运行状况的，其部分信息可以视为是一次运行记录  
类的类图如下：

| Bus  |
|--|
| - id: int<br>- detime: int<br>- arrtime: int<br>- state: int<br>- can: bool  |
| + Bus()<br>+ returnId(): int<br>+ returnDetime(): int<br>+ returnArrtime(): int<br>+ returnState(): int<br>+ returnCan(): bool<br>+ changeId(): int<br>+ changeDetime(): int<br>+ changeArrtime(): int<br>+ changeState(): int |

类的函数成员说明：

| 函数成员                       | 功能              |
|----------------------------|-----------------|
| Bus();                     | 无参数构造函数，用于新建公交车 |
| int returnId();            | 获取车辆代号信息        |
| int returnDetime();        | 获取车辆出发时间信息      |
| int returnArrtime();       | 获取车辆返回时间信息      |
| int returnState();         | 获取车辆所运行的线路      |
| bool returnCan();          | 获取车辆能否正常运行的信息   |
| void changeId(int i);      | 更新车辆代号信息        |
| void changeDetime(int d);  | 更新车辆出发时间信息      |
| void changeArrtime(int a); | 更新车辆到达时间信息      |
| void changeState(int s);   | 更新车辆所运行的线路      |

类的数据成员说明：

| 数据成员                | 含义         |
|---------------------|------------|
| <b>int id;</b>      | 记录车辆代号信息   |
| <b>int detime;</b>  | 记录车辆出发时间信息 |
| <b>int arrtime;</b> | 记录车辆返回时间信息 |
| <b>int state;</b>   | 记录车辆所运行的线路 |
| <b>bool can;</b>    | 记录车辆能否正常运行 |

## 2. 位置坐标类：Position

该类是用于记录某点位置坐标信息的，并执行有关位置信息的一些操作类的类图如下：

| Position   |
|--|
| - x: double<br>- y: double   |
| + Position()<br>+ Position(double initX, double initY)<br>+ getX(): double<br>+ getY(): double<br>+ distance(Position &p): double<br>+ changeX(double newX): void<br>+ changeY(double newY): void<br>+ printPosition(): void |

类的函数成员说明：

| 函数成员   | 功能                 |
|--|--------------------|
| <b>Position();</b>                           | 无参数构造函数，用于新建位置坐标   |
| <b>Position(double initX, double initY);</b> | 带参数构造函数，用于载入数据     |
| <b>double getX();</b>                        | 返回横坐标              |
| <b>double getY();</b>                        | 返回纵坐标              |
| <b>double distance(Position &amp;p);</b>     | 获取当前位置与另一个空间点的直线距离 |
| <b>void changeX(double newX);</b>            | 更改横坐标              |
| <b>void changeY(double newY);</b>            | 更改纵坐标              |
| <b>void printPosition();</b>                 | 打印位置坐标             |

类的数据成员说明：

| 数据成员             | 含义       |
|------------------|----------|
| <b>double x;</b> | 记录位置的横坐标 |
| <b>double y;</b> | 记录位置的纵坐标 |

### 3. 公交车站类：BusStop (继承自 Position 类)

该类是用于记录公交车站的基本信息的

类的类图如下：

| BusStop   |
|---|
| - name: std::string<br>- lines: std::vector<int>  |
| + BusStop()<br>+ BusStop(std::string s, double inix, double iniy, std::vector<int> ss):Position(inix, iniy)<br>+ getName(): std::string<br>+ changeName(std::string newN) : void<br>+ changePos(double newX,double newY) : void<br>+ printInfo() : void |

类的函数成员说明：

| 函数成员  | 功能               |
|---|------------------|
| BusStop();  | 无参数构造函数，用于新建公交车站 |
| BusStop(std::string s, double inix, double iniy, std::vector<int> ss):Position(inix, iniy); | 带参数构造函数，用于载入数据   |
| std::string getName();  | 获取公交车站名称信息       |
| void changeName(std::string newN);  | 更改公交车站名称信息       |
| void changePos(double newX,double newY);  | 更改公交车站位置坐标信息     |
| void printInfo();   | 输出公交车站的基本信息      |

类的数据成员说明：

| 数据成员                    | 含义            |
|-------------------------|---------------|
| std::string name;       | 记录公交车站名称信息    |
| std::vector<int> lines; | 记录公交车站经过的线路信息 |

### 4. 线路类：Line

该类是用于记录线路的基本信息的

类的类图如下：

| Line   |
|--|
| <ul style="list-style-type: none"> <li>- id: int</li> <li>- total_distance: double</li> <li>- distances: std::vector&lt;double&gt; distances</li> <li>- index: int</li> </ul>  |
| <ul style="list-style-type: none"> <li>+ Line()</li> <li>+ Line(int Id, std::vector&lt;double&gt; dd, int in)</li> <li>+ TotalD(): double</li> <li>+ TotalS() : double</li> <li>+ ind(): int</li> <li>+ returnId(): int</li> <li>+ returnDiss(): std::vector&lt;double&gt;</li> <li>+ D(int in): double</li> <li>+ printInfo():void</li> </ul> |

类的函数成员说明：

| 函数成员   | 功能                       |
|--|--------------------------|
| <b>Line();</b>   | 无参数构造函数，用于新建线路           |
| <b>Line(int Id, std::vector&lt;double&gt; dd, int in);</b> | 带参数构造函数，用于载入数据           |
| <b>double TotalD();</b>                                    | 获取线路总长信息                 |
| <b>int TotalS();</b>                                       | 获取线路涉及的公交车站总数信息          |
| <b>int ind();</b>  | 获取某车站在线路上的编号信息           |
| <b>int returnId();</b>                                     | 获取线路代号信息                 |
| <b>std::vector&lt;double&gt; returnDiss();</b>             | 获取线路各段的长度信息              |
| <b>double D(int in);</b>                                   | 获取 source 车站到某一车站 in 的距离 |
| <b>void printInfo();</b>                                   | 输出线路的基本信息                |

类的数据成员说明：

| 数据成员  | 含义              |
|---|-----------------|
| <b>int id;</b>                              | 记录线路编号          |
| <b>double total_distance;</b>               | 记录线路总距离信息       |
| <b>std::vector&lt;double&gt; distances;</b> | 记录线路各相邻车站之间的距离  |
| <b>int index;</b>                           | 记录一个车站在线路上的位置信息 |

## 5. 时间类：Time

该类是用于处理时间的  
类的类图如下：

| Time  |
|---|
| - hr: int<br>- min: int<br>- sec: int   |
| + Time()<br>+ Time(const Time& t)<br>+ Time(int h, int m, int s)<br>+ prinTime(): void<br>+ adv(int inc_h, int inc_m, int inc_s) : void<br>+ early(Time t) : bool<br>+ gap(Time t) : int<br>+ val():int |

类的函数成员说明：

| 函数成员  | 功能                                    |
|---|---------------------------------------|
| <b>Time();</b>                                    | 无参数构造函数，用于新建时间                        |
| <b>Time(const Time&amp; t);</b>                   | 复制构造函数                                |
| <b>Time(int h, int m, int s);</b>                 | 带参数构造函数，用于载入数据                        |
| <b>void prinTime();</b>                           | 按照固定格式输出时间信息                          |
| <b>void adv(int inc_h, int inc_m, int inc_s);</b> | 在原有的时间上增加一个时间段                        |
| <b>void reset();</b>                              | 将时间清零                                 |
| <b>bool early(Time t);</b>                        | 判断时间是否更早于另一个时间                        |
| <b>int gap(Time t);</b>                           | 计算两个时间点的时间差，以分钟计                      |
| <b>int val();</b>                                 | 以 0 时 0 分 0 秒为基准，返回时间的绝对大小表征时间早晚，以分钟计 |

类的数据成员说明：

| 数据成员            | 含义        |
|-----------------|-----------|
| <b>int hr;</b>  | 记录时间的小时信息 |
| <b>int min;</b> | 记录时间的分钟信息 |
| <b>int sec;</b> | 记录时间的秒信息  |

## 6. 用户信息类：UserInfo

该类是用于记录用户的用户名和密码等信息并进行封装操作

类的类图如下：

| UserInfo   |
|--|
| - UsrName : std::string<br>- Psword : std::string  |
| + UserInfo()<br>+ UserInfo(std::string name, std::string pw)<br>+ getName(): std::string<br>+ getPw(): std::string |

#### 类的函数成员说明：

| 函数成员   | 功能             |
|--|----------------|
| <b>UserInfo();</b>                                 | 无参数构造函数，用于新建用户 |
| <b>UserInfo(std::string name, std::string pw);</b> | 带参数构造函数，用于载入数据 |
| <b>std::string getName();</b>                      | 获取用户的用户名信息     |
| <b>std::string getPw();</b>                        | 获取用户的密码信息      |

#### 类的数据成员说明：

| 数据成员                        | 含义         |
|-----------------------------|------------|
| <b>std::string UsrName;</b> | 记录用户的用户名信息 |
| <b>std::string Psword;</b>  | 记录用户的密码信息  |

## 二、 主要技术难点

首先是信息组织形式的选择。考虑到本程序的定位是公交调度和管理系统，并面向乘客开放一定的信息，而公交系统本身就是比较复杂的系统，故而在信息的组织形式上遇到了一定的困难。若是在每次运行程序时都要从键盘输入线路信息，用户的操作性必然会非常差，同时考虑到公交系统的线路网络信息更新不那么频繁，故而采用文件来存储线路和公交车站的相关信息，通过读写文件来支持一系列操作。整个程序使用了如下几个文件：

| 文件             | 存储形式   |
|----------------|--|
| BustopInfo.txt | 每四行分为一个板块，分别为公交车站的名<br>称，横坐标，纵坐标，经过的各条线路                     |
| LineInfo.txt   | 第一行是线路网络的总线路条数，接下来每<br>三行分为一个板块，分别为线路的 id，历经<br>的车站，各车站之间的距离 |
| config.txt     | 第一行为车辆运行速度，第二行为车头时<br>距，第三行为系统开始运行时间，第四行为<br>系统终止运行时间        |
| record.txt     | 每一行是一条行驶记录，分别为线路代号、<br>车辆代号、出发时间、返回时间                        |
| UserInfo.txt   | 每两行分为一个板块，分别为用户名和密码  |

但是线路和公交车站之间的关系是比较复杂的,在进行更新和删除等修改操作时需要将文件做部分内容的更新,这部分内容在 C++提供的基本的文件 I/O 操作中是找不到的。故而在实际操作时新建了存储有线路信息和公交信息的文档的备份,以实现边读边修改边写的功能。

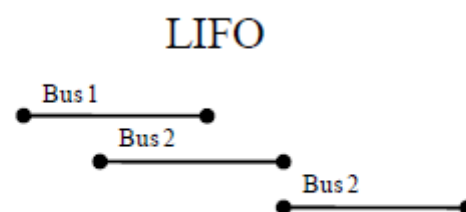
其次是数据结构的选择。考虑到在读取文件时比较容易获取 `string` 对象,因此程序中很多地方都采用了 `string` 对象来存储数据。而线路是天然的线性结构,因此使用数组、链表或者 C++标准模板库提供的一些容器都可以,为了方便性选用了 `vector`。但是在定义一些函数时将 `vector<string>` 作为函数参数出现了一些问题,通过间接定义函数的方式避免了这个问题。另外,在公交调度算法中使用了 C++标准模板库提供的栈和队列等结构。

而后是公交调度算法的实现,之前在一门专业课接触了一些调度算法,对于其原理也有比较清晰的认识,但是实际实现中还是遇到了一些困难,不过最终还是成功实现了 LIFO 公交调度算法。

### 三、 算法设计

由于很多操作都是基于文件读取实现的,而基础的文件读取则是从文件头逐行读到文件尾,故而在查找过程中都是采用的对于文件内容的遍历性质的顺序查找。这样做的效率比较低,但是考虑到大规模的公交系统包含的线路条数和公交数也是比较有限的,故而程序整体的响应没有问题。在使用 `vector` 进行操作时,大部分情况时输出全部信息,也是遍历性质的操作。

公交调度算法如下图所示



车辆按照顺序被派出,而后依次返回。当有新的需求发生时,最后回来的车辆响应该需求被派出。

### 四、 不足之处

- 1) 在执行一些操作时首先需要获取配置文件中的部分内容,考虑到无法有效将这些内容以较基本的数据结构组织起来,故而读取文件操作较为频繁,部分代码没有得到有效的复用;
- 2) 线路和公交车站之间的相互关系较为复杂,现在导入相关信息都是通过文件形式,虽然通过一系列函数来保证数据的一致性,但若是文件的存储形式发生问题,整个系统数据的一致性将会被破坏;
- 3) 事实上,程序处理的是仅有一个公交总站的简单情形,配置文件中的描述公交系统的信息也是比较简单的情况;
- 4) 为乘客提供出行规划意见采用的是一种非常简单的策略,并不能保证出行规划线路是最短的。

# 用户手册

本程序主要针对两类用户群体，分别为公交系统的管理者和使用者。两者的操作权限不同，不过两种用户都可以在程序运行时根据程序的文字提示进行操作。具体的使用说明见下文：

1. 启动程序
2. 根据程序界面提示选择管理员操作或是乘客操作
  - 1) 若选择管理员操作，需要进一步确认是采取登录操作还是注册操作
    - a) 若选择登录操作，后续操作如下：  
根据程序界面提示输入管理员帐号；  
根据程序界面提示输入管理员密码；  
当帐号和密码都正确后操作界面将出现一个主菜单，用户可以通过选择所要操作对应的数字来实现该操作。
    - b) 若选择注册操作，后续操作如下：  
根据程序界面提示输入要注册的管理员帐号；  
根据程序界面提示输入要注册的管理员密码；  
信息会被记录，操作界面将出现一个主菜单，用户可以通过选择所要操作对应的数字来实现该操作。

以下介绍管理员的操作面板所涉及的操作：

## A. 查询线路网络概况

选择该操作后，用户界面将列出所有的公交线路的信息，包括线路网络包括的总线路条数以及各条线路的线路代号、包括车站的总数、历经的各个车站的名称、相邻车站之间的线路距离和线路的总长度等信息。

## B. 查询公交车站的基本信息

选择该操作后，用户界面将要求用户输入要查询的公交车站名称。用户输入之后回车，界面会列出要查询的公交车站的信息，包括车站名称、位置坐标和历经的线路等信息。

## C. 查询公交车站的运行时刻表

选择该操作后，用户界面将要求用户输入要查询的公交车站名称。用户输入之后回车，界面会列出要查询的公交车站所经过的每条线路在两个不同方向上的运行时刻表，运行时刻表每一行都是特定线路特定运行方向的车辆在所查询车站的到达时间。

## D. 查询公交车的整体运行状况

选择该操作后，用户界面将列出整个公交系统运行所需要的最小的公交车数目以及每个公交车一天需要运行的次数。

## E. 查询单个公交车的运行状况

选择该操作后，用户界面将要求用户输入要查询的公交车代号。用户输入之后回车，界面会列出所查询公交车在当前时间在哪条线路上运行，以及该公交车每天的运行次数以及分别在不同线路上的运行次数。

## F. 删除公交线路

选择该操作后，用户界面将要求用户输入要删除的线路代号。用户输入之后回车，界面会列出因为删除线路而所需要删除的公交车站名称，成功删除之后会给出提示。记载有线路和公交车站信息的文件也会更新。

## G. 增加公交线路

选择该操作后，用户界面将要求用户确认是否要将指定文件中的线路信息导入到现



有的系统之中。如果要增加新的线路选择 Y，否则选择 N。用户确认之后会进行一系列操作，更新相关文件中的信息，并提示操作成功；否则界面会提示用户已经取消了该操作。

#### H. 退出程序

选择该操作后系统将在保存完当前数据后退出程序。

2) 若选择乘客操作，操作界面将直接出现一个主菜单，用户可以通过选择所要操作对应的数字来实现该操作

以下介绍乘客的操作面板所涉及的操作：

- A. 查询线路网络概况（同管理员）
- B. 查询公交车站的基本信息（同管理员）
- C. 查询公交车站的运行时刻表（同管理员）
- D. 获取距离自己所在地最近的公交车站名称

为了方便操作，程序在运行时会随机生成乘客的所在地和目的地的位置坐标并保证坐标位于线路网络覆盖范围内。选择该操作后，程序会返回乘客所在地的位置以及距离自己最近的公交车站的名称。

- E. 获取距离自己目的地最近的公交车站名称

选择该操作后，程序会返回乘客目的地的位置坐标以及距离自己目的地最近的公交车站的名称。

- F. 提供出行规划意见

选择该操作后，界面会提供路径规划和换乘等意见。

- G. 退出程序（同管理员）

## 测试用例

首先展示管理员的测试用例，具体如下所示：

- A. 进入程序之后，会提供不同的身份选择

```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe
*****
*      欢迎使用      *
*****

请选择你的身份（1-管理员 2-乘客）： _
```

- B. 键入 1 选择管理员身份，会提供登录和注册操作选择

```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe
*****
*      欢迎使用      *
*****

请选择你的身份（1-管理员 2-乘客）： 1

请选择接下来的操作（1-登录 2-注册）：
```

- C. 键入 1 选择登录，会提示输入用户名和密码，认证成功后会进入主操作界面，并提示后续操作

输入用户名：admin

输入密码：admin（实际操作中输入密码不显示）

```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe
请选择接下来的操作（1-登录 2-注册）： 1
请输入用户名：admin
请输入密码：

登录成功！

您可以选择执行以下操作
-----
1  查询线路网络概况
2  查询公交车站的基本信息
3  查询公交车站的运行时刻表
4  查询公交车的整体运行状况
5  查询单个公交车的运行状况
6  删除公交线路
7  新增公交线路
8  退出程序
-----

请选择接下来的操作(输入操作之前的数字即可)：
```

- D. 选择操作 1, 查询线路网络概况 (受篇幅所限, 只截了前两条线路的信息)

```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe

请选择接下来的操作(输入操作之前的数字即可): 1

线路网络共包含有5条线路, 各条线路的大致信息如下

线路代号:1
历经车站分别为:
source second third fourth fifth
线路共包含5个车站
相邻车站之间的距离依次为:
1.25 2.01 1.68 2.22
线路总长为7.16千米

线路代号:2
历经车站分别为:
source apple pear peach watermelon banana
线路共包含6个车站
相邻车站之间的距离依次为:
1.64 2.36 3.02 0.98 1.12
线路总长为9.12千米
```

- E. 选择操作 2, 查询公交车站的基本信息

输入查询车站的名称: source

```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe

请选择接下来的操作(输入操作之前的数字即可): 2
请输入要查询的车站名称: source

source 车站信息
位置: (0,0)
经过线路: 1 2 3 4 5
```

- F. 选择操作 3, 查询公交车站的运行时刻表 (受篇幅所限, 只截一个方向运行时刻表的部分信息)

输入查询车站的名称: tiger

```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe

请选择接下来的操作(输入操作之前的数字即可): 3
请输入要查询时刻表的车站名称: tiger
该公共车站有1条线路通过

线路4在tiger车站的运行时刻表

从source到终点站的时刻表如下:
07:26:00
07:38:00
07:50:00
08:02:00
08:14:00
08:26:00
08:38:00
08:50:00
09:02:00
```

- G. 选择操作 4，查询公交车的整体运行状况

```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe

请选择接下来的操作(输入操作之前的数字即可): 4
支持整个公交系统正常运行需要19辆公交车
每天在每条线路上公交车要行驶83个来回
```

- H. 选择操作 5，查询单个公交车的运行状况

输入查询公交车的名称: 1

```
选择D:\THU\2019秋\C++\实验\HW\Debug\HW.exe

请选择接下来的操作(输入操作之前的数字即可): 5
请输入您要查询的公交车代号: 1
1号车现在没有运行
1号车每天运行21个来回
每天在1号线上该公交车要行驶8个来回
每天在2号线上该公交车要行驶0个来回
每天在3号线上该公交车要行驶7个来回
每天在4号线上该公交车要行驶6个来回
每天在5号线上该公交车要行驶0个来回
```

- I. 选择操作 6，删除指定公交线路

输入要删除线路的代号: 4

```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe

请选择接下来的操作(输入操作之前的数字即可): 6
请输入您要删除的线路代号: 4
elephant车站已经被删除!
goat车站已经被删除!
fish车站已经被删除!
tiger车站已经被删除!
4号线已经成功删除!
```

- J. 选择操作 7，新增公交线路

选择 N

```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe

请选择接下来的操作(输入操作之前的数字即可): 7
请确认你已在LineInfo_new.txt文件中完善新增线路的信息?<Y/N>: n
您已取消该操作
```

选择 Y

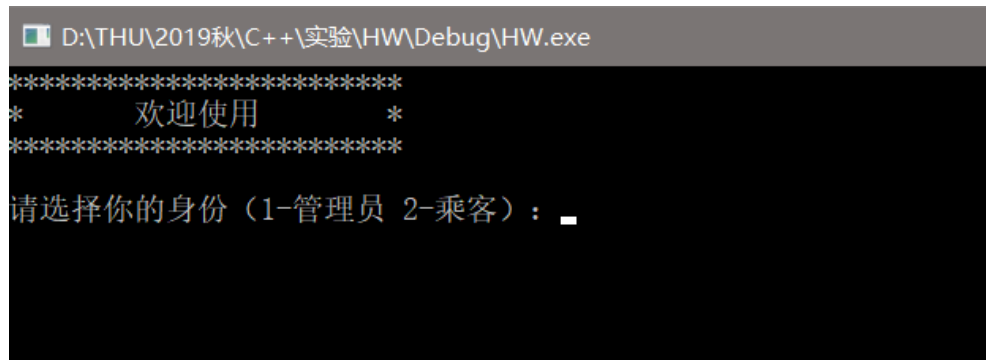
```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe

请选择接下来的操作(输入操作之前的数字即可): 7
请确认你已在LineInfo_new.txt文件中完善新增线路的信息?<Y/N>: y
成功增加5号线
```

- K. 选择操作 8，退出程序，文件自动保存信息

而后展示乘客的测试用例，具体如下所示（前三个操作与管理员相同，不再距离）：

- L. 进入程序之后，会提供不同的身份选择



```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe
*****
*      欢迎使用      *
*****

请选择你的身份（1-管理员 2-乘客）：_
```

- M. 键入 2 选择乘客身份，会直接进入主操作界面，并提示后续操作

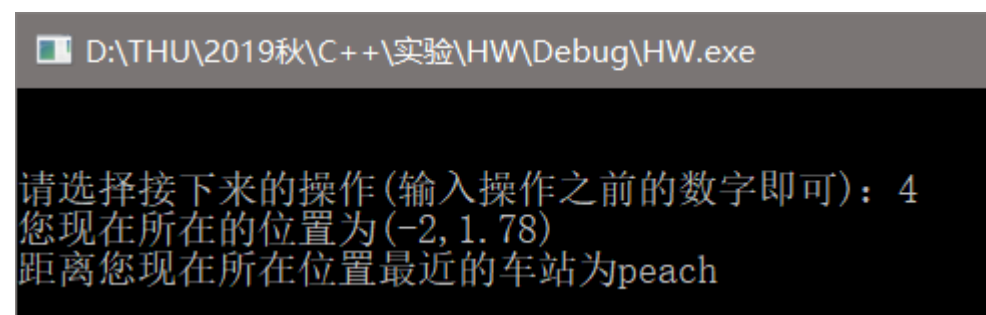


```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe
*****
*      欢迎使用      *
*****

请选择你的身份（1-管理员 2-乘客）：2

您可以选择执行以下操作
-----
1  查询线路网络概况
2  查询公交车站的基本信息
3  查询公交车站的运行时刻表
4  获取距离自己所在地最近的公交车站名称
5  获取距离自己目的地最近的公交车站名称
6  提供出行规划意见
7  退出程序
-----
```

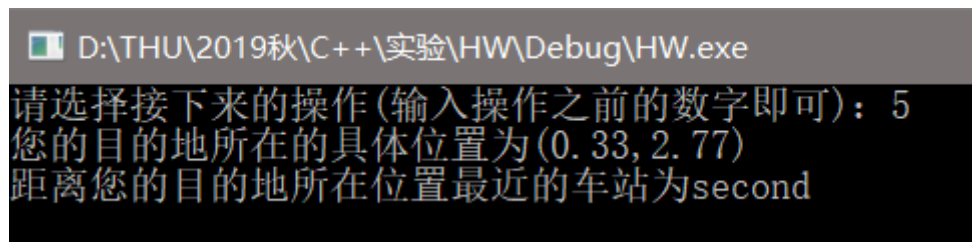
- N. 选择操作 4，获取距离自己所在地最近的公交车站名称



```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe

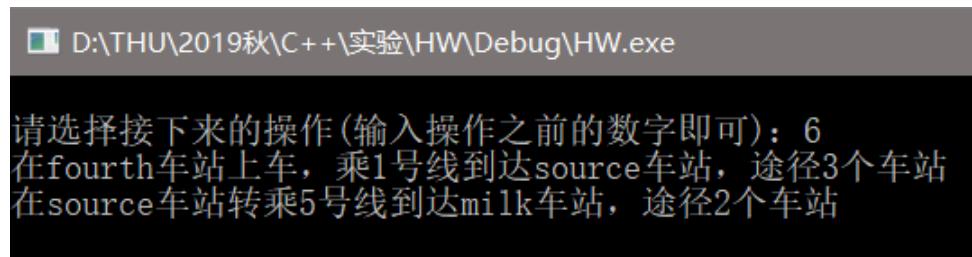
请选择接下来的操作(输入操作之前的数字即可)：4
您现在所在的位置为(-2, 1.78)
距离您现在所在位置最近的车站为peach
```

- O. 选择操作 5，获取距离自己目的地最近的公交车站名称



```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe
请选择接下来的操作(输入操作之前的数字即可): 5
您的目的地所在的具体位置为(0.33, 2.77)
距离您的目的地所在位置最近的车站为second
```

- P. 选择操作 6，获取出行规划意见



```
D:\THU\2019秋\C++\实验\HW\Debug\HW.exe
请选择接下来的操作(输入操作之前的数字即可): 6
在fourth车站上车，乘1号线到达source车站，途径3个车站
在source车站转乘5号线到达milk车站，途径2个车站
```

- Q. 选择操作 7，退出程序，文件自动保存信息