

《创新训练课B》课程设计报告

学院： 机电工程与自动化学院

题目： **深圳地铁自助售票系统**

班级： **自动化1班**

姓名： **张泽斌**

学号： **190320121**

教师： **吴晓军**

上交日期：

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 8.27 | 1.0 |  | 张泽斌 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目 录**

[1 引言 1](#_Toc519846913)

[1.1编写目的 1](#_Toc519846914)

[1.2背景 1](#_Toc519846915)

[1.3定义 1](#_Toc519846916)

[1.4参考资料 1](#_Toc519846917)

[2 任务概述 1](#_Toc519846918)

[3 需求分析 1](#_Toc519846919)

[3.1 用户需求分析 1](#_Toc519846920)

[3.2 运行环境 1](#_Toc519846921)

[4 功能及操作介绍 1](#_Toc519846922)

[4.1 操作 1](#_Toc519846923)

[4.2 功能 1](#_Toc519846924)

[5 系统设计 2](#_Toc519846925)

[5.1 总体架构设计 2](#_Toc519846926)

[5.2 模块分析与设计 2](#_Toc519846927)

[5.3软件结构（流程图） 2](#_Toc519846928)

[6 调试与测试 2](#_Toc519846929)

[6.1调试过程 2](#_Toc519846930)

[6.2测试结果 2](#_Toc519846931)

[7 编程中遇到的问题 2](#_Toc519846932)

[7.1 问题1 2](#_Toc519846933)

[7.2 问题2 2](#_Toc519846934)

[7.3 问题3 2](#_Toc519846935)

[8分析总结与心得体会 2](#_Toc519846936)

# **1 引言**

## 1.1编写目的

通过总结该项目取得的成果，和我个人在这个项目中所学到的知识和技能以及我发现的自身的不足，以报告的形式全面清晰地展现出来，既便于课程教师评审我的课程表现，也有助于我个人能力的提升。

## 1.2背景

通过深圳地铁自助售票系统这个项目了解软件实际开发过程，积累一定的项目经验，巩固C语言的知识，提高自己的代码能力和培养自己的创新实践能力。该项目是基于easyx图形图采用C语言开发的带UI交互界面的程序，通过此次开发可以更好地了解UI开发流程和掌握一定的调试测试能力。这个题目既涉及UI交互，也要求数据管理，可以很好地结合前面所学到的两方面知识，提高自身的技能、能力。

## 1.3定义

·UI：用户界面(User Interface)是指对软件的人机交互、操作逻辑、界面美观的整体设计。

·easyx：EasyX Library for C++ 是针对VC 的一套绘图库，接口简单易用，用起来很像TC 的graphics.h 绘图。

·wchar\_t:。wchar\_t是C/C++的字符类型，是一种扩展的存储方式。wchar\_t类型主要用在国际化程序的实现中，但它不等同于unicode编码。unicode编码的字符一般以wchar\_t类型存储。

·vs:Microsoft Visual Studio（简称VS）是美国微软公司的开发工具包系列产品。

·Bug:程序错误（英语：Bug），是程序设计中的术语，是指在软件运行中因为程序本身有错误而造成的功能不正常、死机、数据丢失、非正常中断等现象。

·链表：链表（Linked list）是一种常见的基础数据结构，是一种线性表，但是并不会按线性的顺序存储数据，而是在每一个节点里存到下一个节点的指针(Pointer)。

·可读性：在编程领域，可读性指的是：人类读者对于源代码的功能意图、流程控制和操作运行是否容易把握。

## 1.4参考资料

《easyx文档》 ——easyx官网

《C语言程序设计》 ——苏小红

《数据结构——链表（C语言实现）》 ——简书

《C语言链表的建立、插入和删除》 ——百度文库

《C语言的头文件和宏定义详解》 ——CDSN博客

《char与wchar\_t(TCHAR)之间的相互转换》 ——CDSN博客

# **2 任务概述**

用vs开发一个有UI交互界面的深圳地铁自助售票系统，要求UI界面精美且交互性强，站点信息与真实深圳地铁站点信息基本匹配，并能根据输入始末站的名字推荐最短路径，包括转站方案等等。

# **3 需求分析**

## 3.1 用户需求分析

·UI界面基本要求:

（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、系统说明四个选项。

（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。

·路径推荐算法要求：

1. 站点信息包括站点名字、站点号、是否中转站、所在线路号；
2. 根据始末站所在路线推荐最短路径，包括以下三种情况：
3. 始末站在同一线路上：此时直接推荐打印出直达路线；
4. 始末站不在同一线路：此时有两种方案，一为只转一次站（深圳地铁绝大多数线路都可以直接转到其余路线），二为转两次站。分别算出两种方案的最短路线并打印出来供用户自行选择；

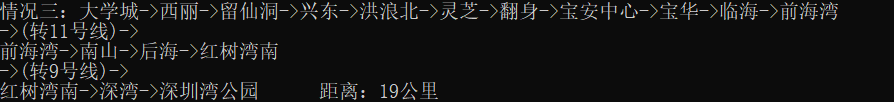
效果大致如下：

以大学城到深圳北站为例：

批注 2020-08-29 161414

以大学城到深圳湾公园为例：

批注 2020-08-29 161448



（3）打印出的推荐路线都显示其单人票价和距离。

·地图查询要求：

（1）可以查看深圳地铁总路线图，也可以查看单条线路路线图。

## 3.2 运行环境

Windows操作系统

# **4 功能及操作介绍**

1. ：旋钮，在开始界面时作为”home”键，点击进去主菜单；在其他界面均作为返回键使用；
2. 开始界面：显示作者及版权；点击旋钮进入主菜单；



1. 系统主菜单：点击相应按钮选择购票、地图查询、退出系统、使用说明等功能；点击旋钮返回开始界面；



1. 购票界面：



点击相应的文本框输入起始站和终点站，点击确定进入下一页（若输入不正确会提示错误并重新输入）；点击旋钮返回；



显示站点信息，点击推荐路线进入下一页，点击旋钮返回；



显示推荐路线，根据需要选择相应路线，点击旋钮返回；



点击文本框输入购票数，默认票数为1，点击“确定购票”进入下一页，点击“取消”返回推荐路线界面，点击旋钮返回主菜单；



购票成功，点击确定或旋钮返回主菜单。

1. 地图查询：点击相应路线查看地图，返回操作同上；

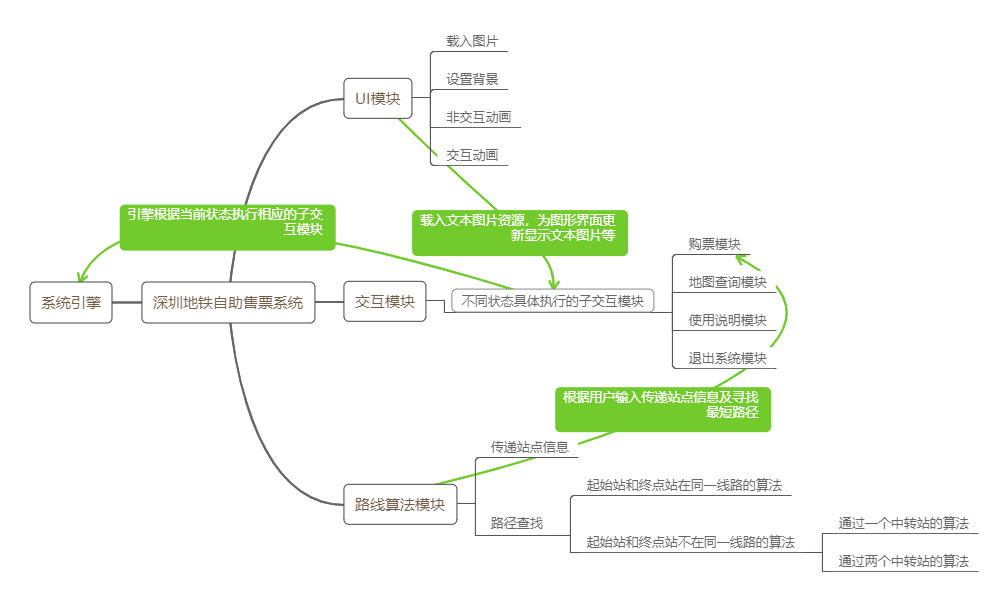


1. 查看使用说明：返回操作同上。



# **5 系统设计**

## 5.1 总体架构设计



该系统分为4部分：系统引擎、UI模块、交互模块、路线算法模块。将各个不同功能的函数封装进相应的模块之中，由不同的子交互模块调用。系统引擎根据当前状态不断循环调用相应的子交互模块，从而使程序运行起来。

## 5.2 模块分析与设计

·UI模块（UI.h & UI.cpp）：

·加载背景图片

int Background();

·显示欢迎和版权信息

int welCome();

·从初始界面到系统主菜单的过渡动画

int MidAnimation();

·从其他子界面到系统主菜单的过渡动画

int MidAnimation\_b(int a);

·主按钮旋转动画

int buttonRo(int a,bool b);

·状态二时显示图片

whenstate\_2();

·状态二按钮特效

int button\_anmi\_2(int a,bool b);

·状态5时的函数

int state\_5();

·状态4时界面画面

int showstate4();

·进入投币界面动画

int coin\_mid(TCHAR per\_pay[],TCHAR all\_pay[],int ticket\_num,int fin\_fee,bool a);

·路线算法模块（routesearch.h & routesearch.cpp）：

算法思路

创建八条Station\_info链表（Station\_info为结构体变量，定义见下文，下文List\_info相同）将八条线路上所有站点的信息录入，再用一条List\_info链表存储八条信息链表的表头，从而存储所有路线的站点信息。

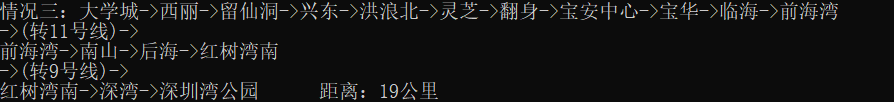
**情况一**：当起始站和终点站在同一线路上时，直接在起始站和终点站所在的路线的信息链表上找到起始站的地址以及起始站到终点站之间的站点数n（有正负），从起始站的地址开始根据n的正负情况向前或向后（即起始站到终点站的方向）遍历直到终点站处停止。在此过程中每过一个节点打印出该节点所录入的站点名称，并打印一个“->”符号，最终定义一个字符串变量最终将整条路径的字符串保存，即可得到如下效果。

批注 2020-08-29 161414

**情况二**：当起始站和终点站不在同一路线上时，寻找跨一次站的路径。遍历起始站和终点站所在的两个链表，找到两站的地址以及所有共同中转站的地址，创建一个Situation\_1\_info（结构体变量，定义见下文）链表存储所有中转站的地址，称为中转站信息表。遍历中转站链表，找到中转站在起始站所在的链表上的地址1以及在终点站所在的链表上的地址2，分别对起始站和中转站地址1、终点站和中转站地址2使用情况一的方法，再将不同中转站方案的路径距离进行比较得到最短路径，最终定义一个字符串变量最终将整条路径的字符串保存，即可得到如下效果。

批注 2020-08-29 161448

**情况三**：当起始站和终点站不在同一路线上时，寻找跨两次站的路径。遍历终点站所在的信息链表，找到所有可能的第二个中转站，创建一个Situation\_2\_info链表存储这些中转站点的地址。遍历第二个中转站的信息链表，对第二个中转站和起始站使用情况二的方法，再对第二个中转站和终点站使用情况一的方法，最后将不同中转站组合的方案进行比较得到最短路径。最终定义一个字符串变量最终将整条路径的字符串保存，即可得到如下效果。



具体的变量和函数声明如下：

·用结构体表示各站点信息

typedef struct station\_info{

char station\_name[100];//站名

int route\_number;//路线号

int station\_number;//站号

bool is\_mid\_station;//是否中转站

int con\_amount[7];//经过的路线

int pre\_distance;//与前一站的距离

int nex\_distance;//与后一站的距离

struct station\_info \*prev;//链表上一节

struct station\_info \*next;//链表下一节

}Station\_info;

·用结构体表示跨一个站的一种路线可能，并最终建立链表

typedef struct situation\_1\_info {

Station\_info \*begstation;//出发站地址

Station\_info \*mid\_station;//中转站所在链表地址

Station\_info \*finstation;//终点站地址

int distance;//该种情况总距离

struct situation\_1\_info \*pNext;//下一个节点

}Situation\_1\_info;

·当跨两个站时，使用结构体链表保存所有可能的第二个中转站点的信息

typedef struct situation\_2\_info {

int distan;//选择这个第二中转站时，前半段的最小距离

Station\_info \*station;

struct situation\_2\_info \*pNext;//下一个节点

}Situation\_2\_info;

·用结构体存储各个站点信息链表的表头

typedef struct list\_info {

Station\_info \*head;

struct list\_info \*pNext;//下一个节点

}List\_info;

·各个链表的表头

List\_info \*All\_head;

Station\_info \*head\_1;

Station\_info \*head\_X;

·各个链表长度

int len\_1;

int len\_X;

·各种路线推荐路线字符串

char \*No\_1str;

char \*No\_2str\_a,\*No\_2str\_b,\*No\_2str\_All;

char \*No\_3str\_a,\*No\_3str\_b,\*No\_3str\_c,\*No\_3str\_All;

·各种路线距离

int One\_dis;

int Two\_dis;

int Three\_dis;

·起始站所在路线

int fir\_sta\_num;

·终点站所在路线

int las\_sta\_num;

·初始化始末站的结构体

Station\_info\* Fir\_station;

Station\_info\* Las\_station;

·主函数

int secondmain(const char\* bestation, const char\* finstation);

·情况一：始末站在同一线路上

int one\_stiuation(Station\_info \* bestation, Station\_info \* finstation);

·情况二：中途中转一次

Situation\_1\_info \*two\_stiuation(Station\_info \* bestation, Station\_info \* finstation, Situation\_1\_info\*Shead);

·情况三：中途中转两次

int three\_stiuation(const char\* bestation, const char\* finstation);

·打印路线

int print\_route\_1(Station\_info \* bestation, const int n);

·情况二时找到所有中间站

Situation\_1\_info \*getashead(Station\_info \* bestation, Station\_info \* finstation);

·第三种情况下，找到前半段中最小的可能性

Situation\_1\_info\* Getshortest(Station\_info\* fir\_station, Station\_info\* sec\_station);

·建立链表

Station\_info \*creat\_link1();

Station\_info \*creat\_linkX();

·遍历链表

Station\_info \*check\_one\_List(int len, Station\_info \*head, const char \*station\_name);

Station\_info \*check\_all\_List\_for\_one\_name(const char \*station\_name);

·根据线路号和名字找到在链表的地址

Station\_info \*check\_List\_by\_num\_nam(Station\_info \*in\_station, char \*pPtr);

·当情况三时，建立一个链表存储所有可能的第二个中转站的信息

Situation\_2\_info\*get\_possibility(int n, Station\_info \*head);

·输入中转站数据

int BaoanCenter(int len,Station\_info \*head, const char \*firstname);//宝安中心

int QianHaiWan(int len, Station\_info \*head, const char \*firstname);//前海湾

······

## **·交互模块（whenfunction.h & whenfunction.cpp）**:

·定义一个全局变量记录当前状态，1为开始界面，2为系统主界面，3为购票界面，4为地图查询界面，5为系统说明

int State ;

·执行状态1即初始界面的函数

int whenState\_1();

·执行状态2即初始界面的函数

int whenState\_2();

·执行状态3即使用说明的函数

int whenState\_3();

·执行状态4即使用说明的函数

int whenState\_4();

·执行状态5即使用说明的函数

int whenState\_5();

## **·系统引擎（usubw.cpp）**:

·定义一个全局变量记录当前状态，1为开始界面，2为系统主界面，3为购票界面，4为地图查询界面，5为系统说明

int State ;

·主函数

int main()

{ // 绘图窗口初始化

initgraph(1180, 800);

Background();

do {

if (State == 1)// 当State为1时，执行此与开始界面对应的交互函数

{

whenState\_1();

}

else if (State == 2)// 当State为2时，执行此与系统主界面对应的交互函数

{

whenState\_2();

}

else if (State == 3)// 当State为3时，执行此与购票界面对应的交互函数

{

whenState\_3();

}

else if (State == 4)// 当State为4时，执行此与地图查询界面对应的交互函数

{

whenState\_4();

}

else if (State == 5)// 当State为5时，执行此与使用说明界面对应的交互函数

{

whenState\_5();

}

else if (State == 0)// 当State为0时，退出引擎主循环

{

break;

}

} while (1);

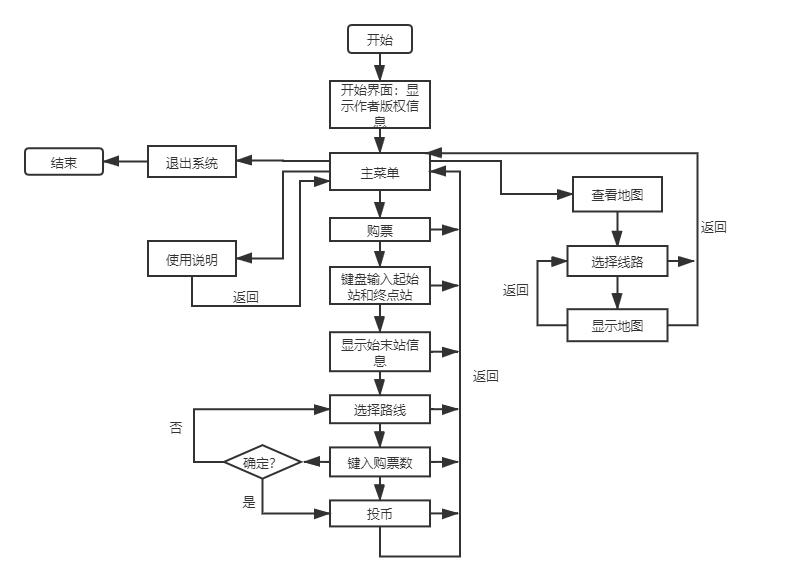
// 关闭绘图窗口，退出程序

EndBatchDraw();

closegraph();

}

## 5.3软件结构（流程图）



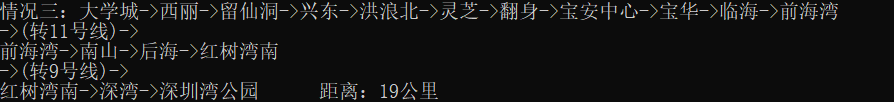
# **6 调试与测试**

## 6.1调试过程

·大学城——>深圳北站

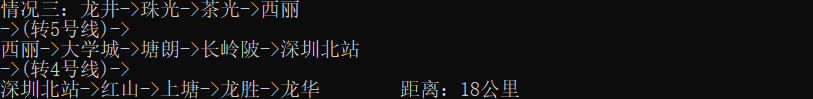
批注 2020-08-29 161414

·大学城——>深圳湾公园

批注 2020-08-29 161448

·龙井——>龙华

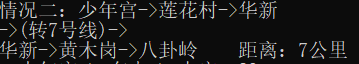
批注 2020-08-30 205449

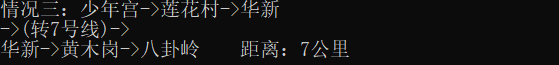


·茶光——>上沙

批注 2020-08-30 205614

·少年宫——>八卦岭





## 6.2测试结果

各个模块运行情况基本良好，完成任务要求。

# **7 编程中遇到的问题**

## 7.1 相邻站点之间的距离无法找到真实数据。

**解决方案**：采用合理范围内的随机数据代替真实数据。

## 7.2 easyx函数库中部分函数字符串参数的编码类型为Unicode编码，不接受char型的字符串参数

**解决方案**：将char型字符串转化为wchar\_t型再进行传参，将转化字符串类型的代码进行函数封装，提高编程便利性和代码可读性。

## 7.3 一个中转站在不同线路的链表中地址不同，对同一中转站不同地址进行操作，结果也不同，实际操作时可能出现错误。如对情况一进行处理时，需要首尾两站的地址在同一链表上，在情况二中，对起始站和中转站使用情况一的操作时，中转站传入的地址如果不是起始站的地址，则无法得到正确结果。

**解决方案**：优化细节，在可能发生此类错误处添加相应代码预处理，避免此类错误的发生。

# **8分析总结与心得体会**

此次课程设计令我收益良多。这个项目让我真正体会到了调试和测试对于程序的重要性，会编码仅仅只是会编程的一部分，要让软件发挥作用，调试和测试是必不可少的步骤。程序的调试修改也不仅仅是按着开发工具的报错提醒去修改那么简单，编译通过也不意味着代码没有bug，有很多错误是通过不断地测试找出来以及改正的。有时候编写一段代码可能只要十分钟，但是找和改bug却需要几个小时。此外，代码的可读性也十分重要，可以提高编程的效率。适当的注释可以防止写到最后连自己都看不懂自己的代码。对我个人而言，这是第一个由我自己从构思到开发到报告一个人完成的完完整整的项目，除了使我获得不少知识和一定的成就感外，更让我对计算机的世界更加好奇和向往。