**总体方案设计说明**

**目录**

[1、软件开发环境： 2](#_Toc12413)

[2、总体结构流程图 2](#_Toc3190)

[3、模块划分 4](#_Toc9691)

[4、数据结构说明和数据字典 5](#_Toc9199)

*[4.1 Form类（主界面）](#_Toc5208)* [5](#_Toc5208)

*[4.2 querywidget类（从界面）](#_Toc26335)* [6](#_Toc26335)

*[4.3 recordwidget类（日志页面）](#_Toc6393)* [6](#_Toc6393)

*[4.4 Widget类（欢迎界面）](#_Toc1742)* [6](#_Toc1742)

*[4.6 navigation （链路规划模块）](#_Toc27987)* [7](#_Toc27987)

*[4.7findTheWay （路由规划模块）](#_Toc13418)* [7](#_Toc13418)

[5、用户使用说明： 7](#_Toc3873)

**1、软件开发环境：**

* 1. 操作系统：建议为Windows10（支持跨平台开发）
  2. 开发框架：Qt 5.3.1版
  3. 版本控制：Git
  4. 编程语言：C++，基于支持C++11及以上标准的版本
  5. 源文件编码：UTF-8（其他编码可能导致中文注释出现乱码的现象）

**2、总体结构流程图**

（图太大，在下一页）

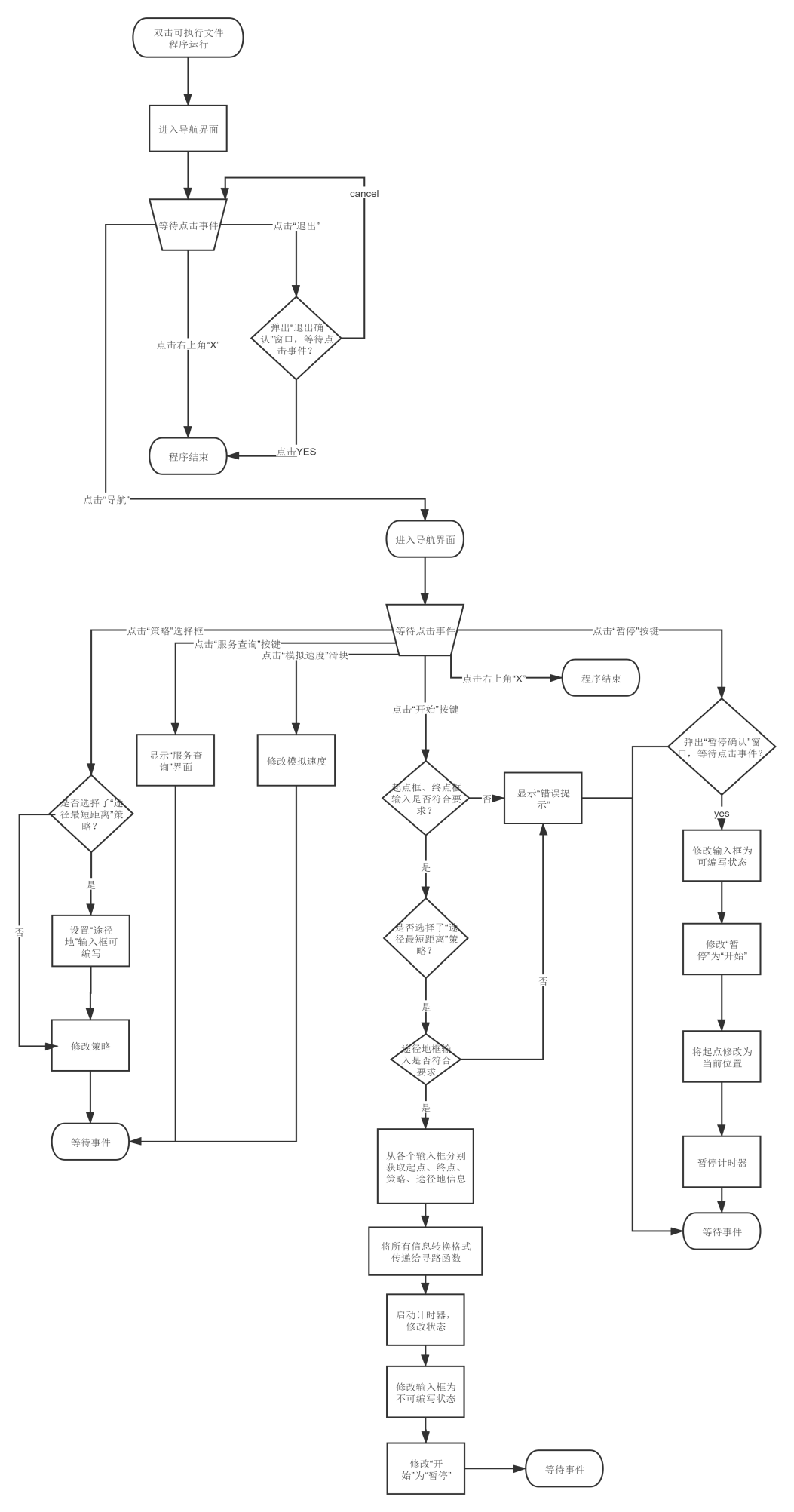


图1 程序总体框架和程序流程图

**3、模块划分**

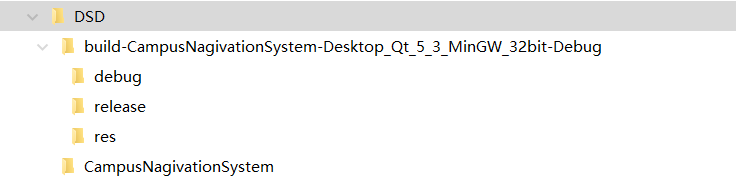


图2 文件夹结构图

|  |  |
| --- | --- |
| 文件 | 说明 |
| DSD | 总文件 |
| debug | 对象文件（.o）以及可执行文件（.exe） |
| res | 存放图片（.png/.jpg）和文本（.txt） |
| CampusNagivationSystem | 头文件（.h）、源文件（.cpp）和软件界面文件（.ui） |

表1 文件夹说明

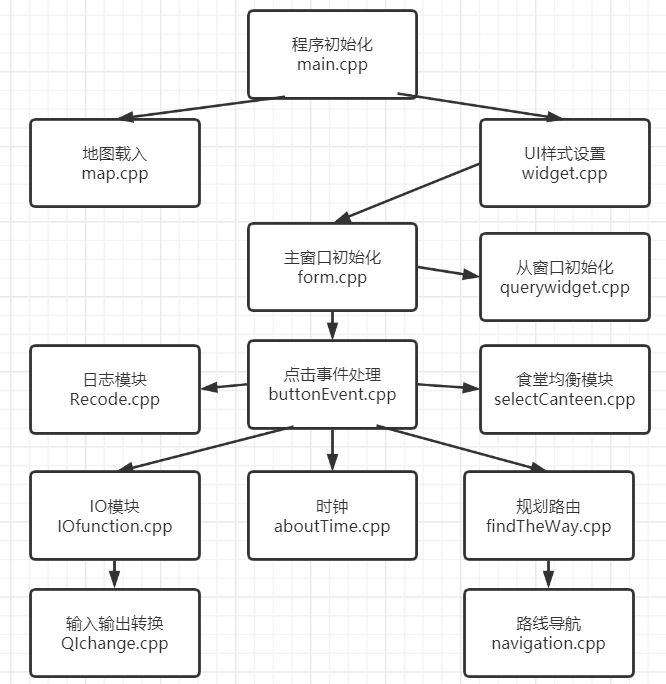


图3 模块划分和调用关系

**4、数据结构说明和数据字典**

***4.1 Form类（主界面）***

public:

explicit Form(QWidget \*parent = 0);//构造函数

void *paintEvent*(QPaintEvent \*);//背景图设置函数

QueryWidget \*query = NULL;//从页面

~*Form*();//析构函数

int origin;//出发地

int terminal;//目的地

int strategy;//导航策略

double actualDistance;//实际距离

double formalDistance;//形式距离

std::vector<int> transfer;//途经地

int pos2int(QString position);//将地点从qstring类型转成int类型

QString int2pos(int positionNumber);//将地点从int类型转成qstring类型

void insertRecord();//日志函数

void selectCanteen();//食堂负载函数

RecordWidget recWid; //日志窗口

private slots: //Qt槽函数，当界面上控件状态改变时被触发

void on\_comboBoxStrategy\_currentIndexChanged(int index); //策略选择

void on\_pushBUttonStart\_clicked(); //点击“开始”

void on\_horizontalSlider\_valueChanged(int value); //模拟倍数

private:

Ui::Form \*ui; //图形化界面操作对象，可获取界面中的输入

int position;//当前位置

double leftTime;//目前抵达终点所需的时间

double speed;//前进速度

std::vector<int> pass\_position;//经过的路径点集

std::vector<int> distance\_of\_change;//剩余距离为何值时速度应该改变

std::vector<double> local\_speed;//在当前位置的前进速度

std::vector<int> getTransfer(QString transfer);//获取UI界面中的的途径地

QTimer \*mytime; //时钟对象

void changeStation();//按照时钟函数被调用，修该状态

void displayStation();//图形显示函数

***4.2 querywidget类（从界面）***

public:

explicit QueryWidget(QWidget \*parent = 0); //构造函数

void *paintEvent*(QPaintEvent \*); //设置背景图

~*QueryWidget*(); //析构函数

void luxian\_show(std::vector<int> t);

private:

Ui::QueryWidget \*ui; //图形化界面操作对象，可获取界面中的输入

***4.3 recordwidget类（日志页面）***

public:

explicit RecordWidget(QWidget \*parent = 0); //构造函数

~*RecordWidget*(); //析构函数

private slots:

//Qt槽函数

void on\_pushButtonClean\_clicked(); //清除日志

void on\_pushButton\_clicked(); //刷新日志

private:

Ui::RecordWidget \*ui; //UI界面操作对象，可获取界面中的输入

***4.4 Widget类（欢迎界面）***

public:

explicit Widget(QWidget \*parent = 0); //构造函数

void *paintEvent*(QPaintEvent \*);//背景图设置函数

Form \* wNag = NULL;//主窗口

~*Widget*(); //析构函数

private:

Ui::Widget \*ui; //UI界面操作对象，在本窗口中几乎无用

***4.5 map模块（地图模块）***

地图结点结构体：

typedef struct{

char node\_name[33]; // 节点名称,不超过31个字符

int next\_node\_number[6]; // 邻接节点的编号列表

int next\_node\_distance[6]; // 到邻接节点的距离

int next\_node\_crowd[6]; // 两节点间道路的拥挤程度

int next\_node\_bicycle[6]; //是否可以通行自行车，1为能，0为不能

} Node;

宏定义：

#define Number\_Total //地图总结点数

#define Number\_Xi //西土城校区结点数

全局变量：

static Node map[Number\_Total]; //地图结点数组

static int distanceBetweenPoint[Number\_Total][Number\_Total]; //最短距离矩阵

函数：

int InitMap(); //载入地图

void warshall(); //计算任意两点间的最短实际距离

void sort\_map(int map\_len); //地图排序

void save\_sorted\_map(QString selected\_map\_name, int map\_len); //保存地图数据

***4.6 navigation （链路规划模块）***

结构体：

typedef struct NaviTreeNode{

int FatherNode; //父节点代号

int formalDisFromStart; //到起点的形式距离

int actualDisFromStart; //到起点的实际距离

} NaviTreeNode; //结果树结点

函数：

bool NotFound(int num, int index, int HaveFound[]); //判断结点是否已经被找到

std::vector<int> navigation(int start, int end, int strategy)；//根据策略返回最佳路径

***4.7findTheWay （路由规划模块）***

函数：

int insert\_gate(std::vector<int> \*points, int strategy, int XS, Form\* ui, int point\_num); //加入校车/公交车等待时间

std::vector<int> join\_path(std::vector<int> points, int strategy, double& actualDistance); //将多段路由拼合成一段

int shortest\_passing(std::vector<int> \*points, int strategy, int start\_index, int end\_index); //路由排序函数

bool is\_station(int one); //判断该结点是否是车站

**5、用户使用说明：**

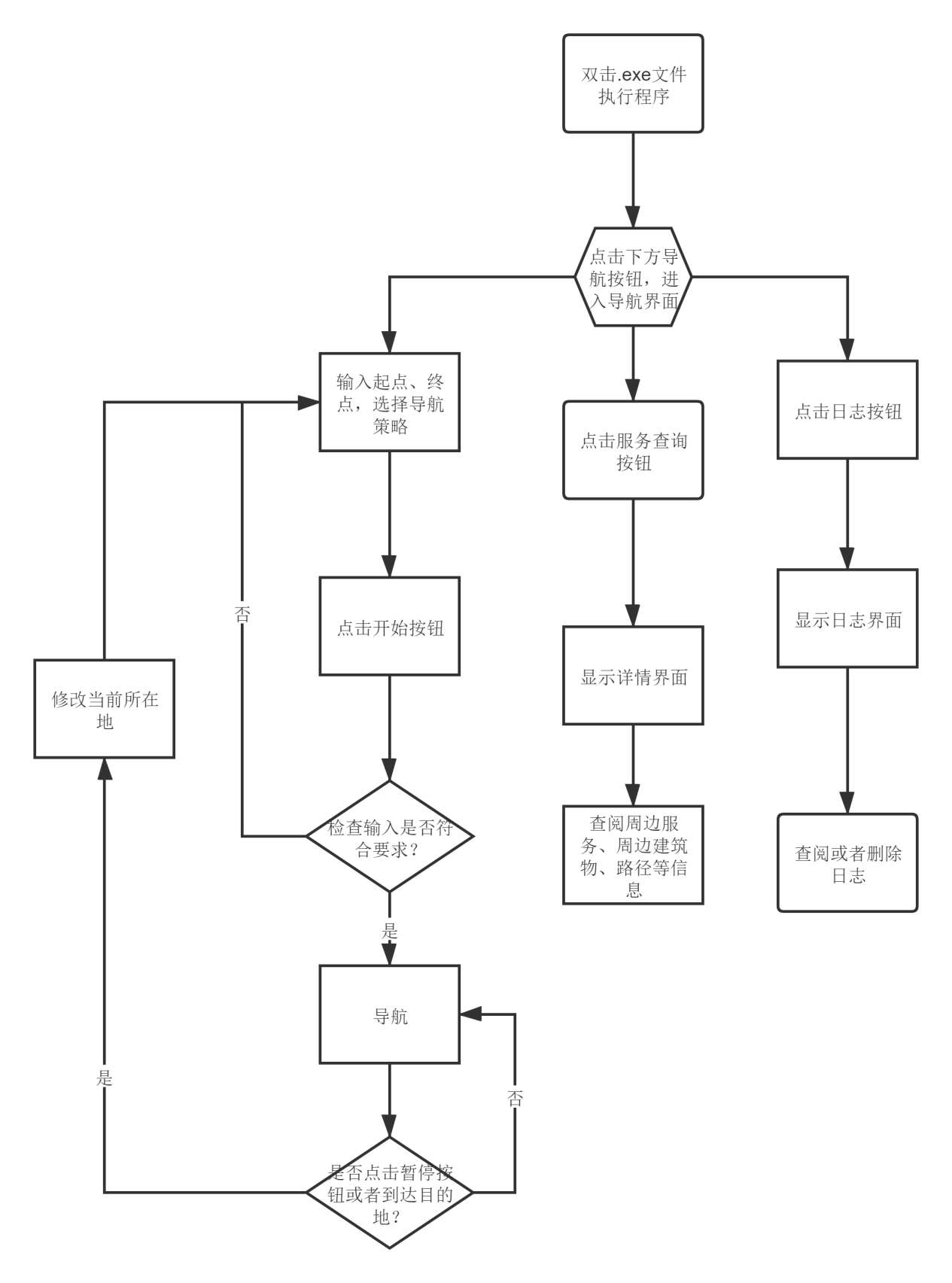


图4 导航软件使用指导图

在图形界面开发中，我们额外编写了许多提示窗口，例如：“文件读取错误”、“地图载入失败”等程序错误提示和“目的地输入错误”、“确认是否暂停导航”等用户使用提示。简洁的用户界面以及丰富的提示信息大大地优化了用户的使用体验。