# 总体设计方案说明

1. **问题整理**

1、呈现三种交通工具：汽车、火车、飞机。

2、某旅客于某一时刻向系统提出旅行要求，系统根据该旅客的要求为其设计一条旅行线路并输出。

3、系统能查询当前时刻旅客所处的地点和状态，包括旅客所停留城市，以及所乘坐的交通工具。

1. **目标功能**

1、初始城市总数为11个。

2、分别建立汽车、火车和飞机的时刻表（航班表），通过文件读取信息初始化时刻表内容，有沿途到站、票价、路程等信息。

3、旅客的输入要求：起点、终点、途经城市，由用户自行选择旅行策略。

1. 提供三种旅行策略：

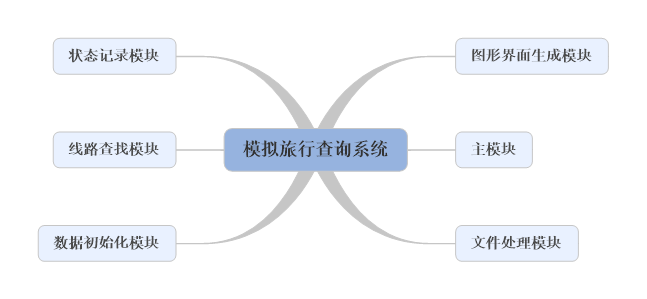
* 最少费用策略：无时间限制，费用最少即可
* 最少时间策略：无费用限制，时间最少即可
* 限时最少费用策略：在规定的时间内所需费用最省

5、旅行模拟查询系统以时间为轴向前推移，打开可执行文件同时系统计时器开始工作，10秒代表1个小时向前推进（非查询状态的请求不计时）；城市内换乘交通工具时间不予考虑，系统时间精确到小时。

6、建立日志文件，对旅客状态变化和键入等信息进行记录。

7、利用QT实现地图界面的可视化，在地图上实时反映出旅客当前的地理位置，交通工具，并且能够动态显示其行进过程。

1. **模块划分**

****

1. **主模块**

模块功能：

主模块负责接收键盘键入命令，包括用户的输入和管理员的输入，分析该命令并调用相应的子模块中的接口函数，并以系统时间为轴向前计时，推进运行。

1. **线路查找模块**

模块功能：

该模块主要负责根据用户输入的起止城市、旅行策略，生成符合条件的旅游线路，该模块是整个系统的核心功能模块。

主要算法：

Dijkstra算法：求一个城市到其他所有城市的最短路径

Floyd算法：求任意两个城市之间的最短路径

算法思路：

根据初始化的文本数据，将读取文件每一个城市抽象为一个节点，生成一个交通路线图，抽象边长为票价/路程的值，前两种策略则根据Floyd算法计算出最短路径生成，第三种策略则是先以时间为权重求得所有路径，然后计算每一条路径的费用，有路径满足，输出，否则，返回“无法满足需求”。

1. **数据初始化模块**

模块功能：

该模块负责初始化载入已有的数据，包括交通工具的时刻表 / 航班表中的数据和登录账户的数据，便于系统的运行。

编写思路：

建立时刻表、账户表的txt，通过行读取文件中的内容，初始化结构中的数据。

1. **状态记录模块**

模块功能：

该模块是系统负责对当前旅客的乘车状态（包括所选交通工具、乘车时间、票价）以及系统所键入信息的记录，便于管理员进行管理。

编写思路：

计时器每十秒钟（代表系统一个小时）刷新一次记录，实时保存当前系统的所有运行数据。

1. **日志文件处理模块**

模块功能：

能够完成相应的对日志文件进行静态写入和查询结果输出操作的功能，方便管理员进行管理操作。

编写思路：

打开状态记录日志文件，可以直接在日志上增添内容。

**（6）图形化界面生成模块**

模块功能：

利用Qt开发框架编写可视化的界面地图，实时显示用户当前的乘车信息，动态的展示旅客的行进路线。

编写思路：

参考相关Qt应用程序开发框架的专业书籍。

1. **结构定义**
2. **枚举相关变量**

enum Vehicle

{Bus = 1,Train = 2,Flight = 3};//枚举交通工具变量值

enum Travel\_Strategy

{MinCost = 1, MinTime = 2, Min\_Cost\_Time = 3};//枚举旅行策略变量值

enum Action

{T\_Login = 1, A\_Login = 2, Travel = 3, Search = 4};//枚举发生事件行为变量值

1. **数据结构定义**

typedef struct{

int mean;//交通方式

int money;//该车次所需价格

int StartTime;//该车次起始时间

int EndTime;//该车次到达时间

int Time;//该车运行时间

}TIME;

typedef struct {

TIME Time[30];//一天最多一条路线十趟车，包含起始时间与到达时间

int i;//路线个数

}Path;

typedef struct {

Path P[NN][NN];

}Graph;

typedef pair <int,TIME> Road;

typedef struct{

int z;

string s;

vector<Road> r;

}Travel;

1. **流程定义**

