旅行模拟项目的设计

一、设计任务的描述

城市之间有三种交通工具（汽车、火车和飞机）相连，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；系统能查询当前时刻旅客所处的地点和状态（停留城市/所在交通工具）。

二、功能需求说明及分析

- 城市总数不少于 10 个
- 建立汽车、火车和飞机的时刻表（航班表）
  - 有沿途到站及票价信息
  - 不能太简单（不能总只是 1 班车次相连）
- 旅客的要求包括：起点、终点、途经某些城市和旅行策略
- 旅行策略有：
  - 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
- 旅行状态查询结果从旅行开始向后推进（10 秒模拟 1 个小时），指示时间（月/日/时）、位置、交通工具等，此查询应可随时退出；
- 不考虑城市内换乘交通工具所需时间，即可以零时间换乘；
- 系统时间精确到小时
- 建立日志文件，对旅客状态变化和键入等信息进行记录
- 选做一：某旅客在旅行途中可更改旅行计划，系统应做相应的操作
- 选做二：用图形绘制地图，并在地图上反映出旅客的旅行过程。

三、总体方案设计说明

本工程使用 visual studio 2015 编写。

整个软件通过接收图形化界面读取的旅客发来的需求，调用相应的算法，之后随着时间的变化对旅客的状态进行更新。

整体模块划分为初始化模块、线路设计模块（包含三种策略）、状态更新模块、图形化模块，其中状态更新模块作为一个独立的线程。

四、数据结构的基本定义

1、数据结构的定义

```
typedef struct//表示具体某辆车的信息
{
  int beginplace;//出发地
  int toplace;//该车的目的地
  int starttime;//发车时间
  int arrivetime;//到达时间
  int money;//票价
  string line;//车次
  string kind;//交通工具种类
}realcar;

typedef struct CarArc//包含两点间的所有车辆
{
```

```
  int pointplace;//该边所指向的顶点
  struct CarArc *nextcar;//指向下一条弧的指针
  //int minarrivetime;//所有车辆中到达目的地时间最早的
  realcar minmoney;// 所有车辆中价格最低的
  int carnum;//这两点间的总车辆数
  realcar truecar[MAXCAR];//所有的车辆
}Car;

typedef struct PlaceNode
{
  string name;//城市名称
  Car *firstcar;//指向该顶点出去的第一辆车
}Place, adjList[VEXNUM+1];

typedef struct
{
  adjList  citymap;//顶点数组
  int places, cars;//记录顶点数和边数
}Graph;

typedef struct//记录系统时间
{
  int year;//年
  int month;//月
  int day;//日
  int hour;//小时
}Timer;

typedef struct
{
  int strategy;//旅游策略
  int name;//记录旅客的编号
  int startplace;//出发地
  int summoney;//所需花费的总费用
  int currentplace;//当前所在的地点
  int state;//更新状态时表示当前状态，1为等车，2为在车上，3为已
到达终点
  int endplace;//目的地
  int starttime;//开始时间
  int limittime;//时间限制
  int arrivetime;//到达当前地点时间huo出发后所花费时间
  Timer startday;//记录出发时的年月日
  int visited[VEXNUM + 1];//该路线已访问过的结点
  realcar  line[VEXNUM + 1];//沿途要乘坐的车
```

```
    int linenum;//表示当前line里有几辆车
  }Traveler;//表示规划出的旅行路线
```

2、数据字典说明

请见文件“数据字典说明.pdf”

五、旅途规划算法设计说明

策略一：

第一个策略是最少费用策略，总体思路是考虑使用 Dijkstra 算法，由于没有时间的限制，整个旅行图的边的权值相对较好确定。为了使查询时更快，我们会事先执行 n 次 Dijkstra 算法，将所得路径存储在一个二维数组 djst[n][n] 中，当需要查询时，我们可以直接在数组中取路径，达到较好的时效性。对于存在途径点的情况，我们分段在 djst 数组中取路径，并将每段的路径组合形成最终的路径。

主要的两个函数：

```
void dijkstra();
Traveler shortmoney(int a, int b)
```

策略二：

这一方案主要的思想是深度优先搜索，如果单纯使用深搜的话复杂度太高，整体查询效果会很慢。最直接我们想到的是在搜索过程中剪枝，剪枝方案为使用一个变量储存当前最小值的阈值信息，只要当前所用时间超过阈值，即减掉这一支，如果最后成功到达目的地则更新阈值信息。

实际操作中，我们在每个点会查找到同一地点的车中，到达时间最早的，加入路线中，在压入栈中；再不断从栈中取数据分析，直至栈空为止。

主要的函数：

```
Traveler mintime(int a, int b,int starttime)//a 为出发地编号，b 为目的地编号
```

策略三：

限时最少费用策略，基本思想依然是深度优先搜索。不过这一次我们不可以只使用时间为阈值信息，因为有了最少费用的要求。我们可以采用双阈值策略，即同时以所规定时间和当前最少费用为阈值，当前路径超过规定时间时舍弃，若超过当前最少费用时也舍去，只有当二者同时满足时才继续搜索下去。若最终可到达最终目的地，则对当前的最少费用进行更新。

这一策略由于需要对每个边做压栈处理，时间消耗较大，为了尽快得出较好的结果，我再搜索是对搜索层数进行了限制，实现了较好的时效性。

主要的函数：

```
Traveler limitmm(int a, int b, int starttime, int limittime)//限时最少费用策略
```

六、状态更新模块设计说明

我们使用全局变量 Timer systime 来记录当前系统的时间，使用 void settime()函数，每隔 10 秒更新一次系统时间，同时在新的时间更新系统中各旅客的状态，同时将状态的变化输出到日志文件中。图形化界面中有用专门的窗口，显示当前的时间，查询旅客状态时，可以在地图上看到旅客的实时状态，并在地图上实时更新。

主要的函数：

```
void settime();
```

七、图形化界面的设计说明

graph.cpp 文件实现图形化界面，用 easyX 编写，需装载 easyX 库，调用头文件 graphics.h 和 conio.h（在头文件 basis.h 中完成）。

实现图形化主要有以下几个函数：

```
void InitBackground();        //初始化背景
void InitMap();               //初始化地图
void InitButton();            //初始化按钮
void InitWin();               //初始化窗口
void Operation();             //用户执行操作
```

InitWin()函数创建一块像素为 1300*700 的画布，并调用 InitBackground()，InitMap()，InitButton()函数对界面进行初始化；

InitBackground()函数打开“背景.jpg”，并贴到画布上；

InitMap()函数打开“地图.jpg”，并贴到画布上；

InitButton()函数初始化各种用户操作按钮、供用户输入及软件需要输出的交互窗口以及在地图上标记出 10 个限定城市；

Operation()函数是实现图形化界面的逻辑主体，在适当的时候调用 InitWin()，InitBackground()，InitMap()，InitButton()实现相关功能，用来对用户的各种操作进行判断及相应。

为了方便进行用户是否可执行某一操作，定义下列开关变量：

```
bool ADDING = 0;                //正在添加旅客
bool ADDTRA_OK = 0;             //旅客添加完成
bool SETSTARTING = 0;           //正在设置出发城市
bool SETSTART_OK = 0;           //出发城市设置完成
bool SETENDING = 0;             //正在设置目的地
bool SETEND_OK = 0;             //目的地设置完成
bool SETTHROUGHING = 0;         //正在设置途经城市
bool SETTHROUGH_OK = 0;         //途经城市设置完成
bool SETSTRING = 0;             //正在选择策略
bool SETSTR_OK = 0;             //策略选择完成
bool ISDEMAND = 0;              //是否查询
bool SETSTARTTIME = 0;          //设置出发时间
bool SETLIMITTIME = 0;          //设置限时最少费用的时间限制
bool SETSTR3 = 0;               //设置策略三的相关时间
```

上述开关变量只有值为 1 或 true 时才可执行相关操作。

为了便于记录旅客的相关信息，设定以下变量：

```
int travnum = 0;          //旅客数量
int timelimit = 0;        //策略三中的时间限制
int starttime = 0;        //旅客旅行开始时间
int startcity = 0;        //旅客出发城市
int endcity = 0;          //旅客目的城市
int strategy = 0;         //旅客所选策略
int seq = 0;              //查询旅客状态时所输入的旅客序号
```

当用户完成一次旅客添加后，上述开关变量进行初始化，以便用户的下次操作。由于本程序实现采用了互斥机制，即用户在进行某一操作时不能进行其

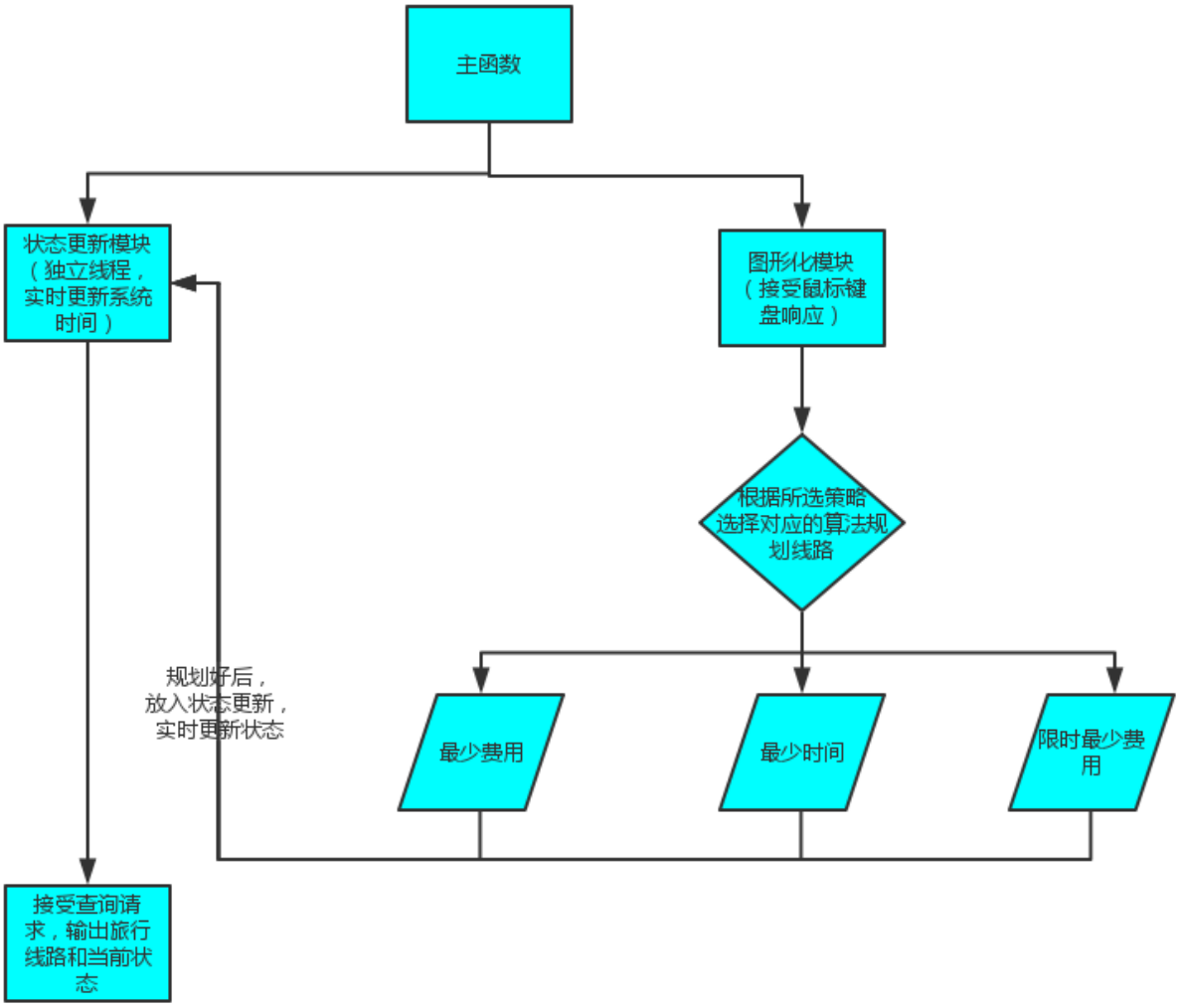
他操作，所以用户不用担心相关开关变量会发生错误变化。

另外，在状态更新线程中会对旅客的状态进行定时刷新

除此之外，用户的每一次操作以及旅客的每一次状态更新都会追加输出到相应的日志文档里，以便查阅。

具体实现请参考 graph.cpp 文档。

八、各模块间的调用关系



九、评价和改进意见

本软件较好的完成了客户需求的各项功能，能够实现在图形化界面的交互设计，旅途规划出的路线也较为合理。

未来在旅客旅途状态的显示上可以做出改进，增加动画效果，使交互体验更好；另外，还可以增加旅行策略的选项，给出更多的选择，来真正满足用户，旅途规划的需求。