各模块设计说明

模块一（TravelSimulationSystem.cpp）

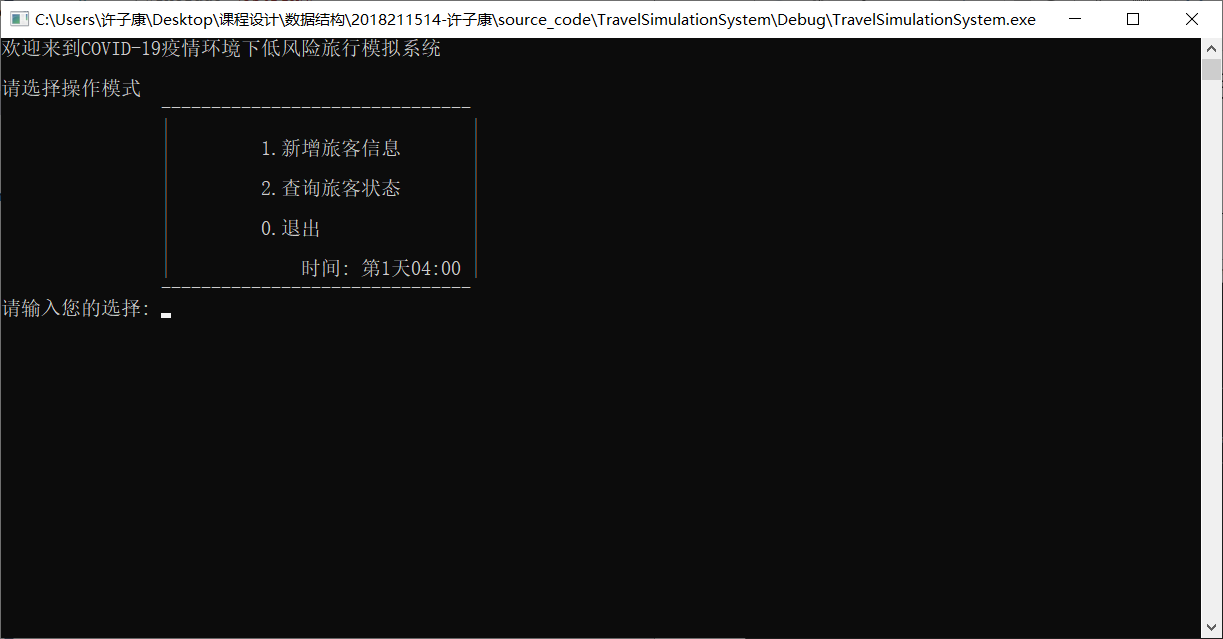
函数：void main(){}

算法：模拟

关系：调用其他模块的内容实现用户操作

功能：进入界面的窗口展示；用户操作选择，并调用function模块中的函数执行用户操作；计算程序运行时间和模拟推进时间





模块二（definitions.h）

函数：无

算法：无

关系：被其他模块调用，全局宏定义

功能：一些宏的定义，定义了CITY\_FILE、ROAD\_FILE、LOG\_FILE、LL和Inf，便于直接使用

模块三（function.h）

函数：void ReadCity();//城市信息读入

void ReadRoad();//交通信息读入

void AddPassenger(int); //新增乘客信息

void QueryPassengerStatus(int); //查询旅客状态

void LogAll(int); //每小时日志写入

算法：无

关系：调用definitions.h模块，定义的函数在function.cpp模块中实现，被主函数调用使用。

功能：定义了上述5个函数，通过主函数调用，function.cpp实现。

模块四（function.cpp）

函数：void Insert(int fr, int to, int tp, int Start)//城市间路线

void AddEdge(int fr, int to, int tp, int Start)//城市内部拆点路线

void ReadCity()//城市信息读入

void ReadRoad()//交通信息读入

int Day(int nowTime)//小时转天

int Hour(int nowTime)//小时转当天的小时

bool Dijkstra(int pId,int fr,int to,int startTime,int endTime)//核心算法，Dijkstra函数求旅客路径

void AddPassenger(int nowTime)// 新增旅客信息

void QueryPassengerStatus(int nowTime)// 查询旅客状态

void LogAll(int nowTime)// 输出信息到日志

算法：堆优化的Dijkstra算法、邻接表、拆点分流

特点：

系统化的实现了不同用户的不同操作，根据实际情况对城市进行拆点操作，一个城市拆分为24个城市，保证了Dijkstra算法的正确性

关系：实现了function.h中定义的函数。

功能：整个工程的核心内容，所有信息的定义和操作的具体实现。

在系统启动前，调用ReadCity和ReadRoad函数读取城市信息和交通信息，ReadCity和ReadRoad分别调用AddEdge和Insert函数，插入城内拆点后的边和城市之间的边。

在用户执行新增旅客信息后，调用AddPassenger函数，执行新增旅客操作。在为旅客规划路径时，调用Dijkstra函数，利用堆优化的Dijkstra算法，找打最佳路径并存储到旅客信息中。

在用户执行查询旅客状态后，调用QueryPassengerStatus()函数，查询旅客状态。

系统每隔2s，也就是模拟时间的1小时，就会调用一次LogAll函数，把日志写入到文件。