1 数据类型、枚举和常量定义

1.1 数据类型别名

为了提高软件开发效率，提高代码可读性，在程序源代码编写过程中，对关键数据类型提供了别名，具体信息如下：

自定义数据类型	原始数据类型	说明
CityID	int	城市唯一标识符类型
VehID	int	车次唯一标识符类型
TravelID	int	旅行实例唯一标识符类型
Time	int	时间类型
DangerIndex	int	风险值类型
CityName	std::string	城市名称类型
VehName	std::string	车次名称类型
CityList	std::vector<City*>	城市列表类型
Schedule	std::vector<Vehicle*>	时刻表类型
Plan	std::vector<Vehicle*>	旅行计划类型
InstanceList	std::vector<TravelInstance*>	旅行实例列表类型

1.2 枚举类型

为了限定特殊数值的范围，定义以下枚举类型：

枚举类型	枚举值	说明
CityKind	LOW=1, MID=2, HIGH=3	城市风险类型
VehKind	AIRPLANE=1, TRAIN=2, BUS=3	交通工具类型
Request, PlanKind	NOLIMIT=0, LIMIT=1	旅行请求
Status	HANG=-1, WAIT=0, RUN_AIR=1, RUN_TRN=2, RUN_BUS=3, FINISHED=4	旅客状态，用于地图绘制

1.3 常量定义

为提高程序可维护性，定义以下常量：

名称	类型	值	说明
INS_TIME_MAX	Time	240	旅行实例理论最大持续时间
DANGER_MAX	DangerIndex	INT_MAX	风险上限（用于最短路径算法）

CLOCK_FREQ_NORMAL	int	10000	系统常规精度时钟频率
CLOCK_FREQ_MEDIUM	int	1000	系统中精度时钟频率
CLOCK_FREQ_PRECISE	int	100	系统高精度时钟频率

2 核心类定义与数据字典

【注：类的默认构造/析构函数，简单初始化构造函数，以及私有属性的 getter 和 setter 不在下表列出】

2.1 控制层

2.1.1 系统控制模块类 Control（Control.h/.cpp）

1. 职责：是整个系统的核心，负责系统初始化，维护系统运行，管理程序数据，控制数据流动。

2. 类属性

属性名	数据类型	说明
私有		
sysPause	bool	系统时钟状态，true 表示暂停状态，false 表示非暂停状态
sysClock	std::thread*	系统时钟子线程
travelStatusRefresher	std::thread*	系统刷新器子线程
公有		
cityList	CityList	城市列表
schedule	Schedule	时刻表
instanceList	InstanceList	旅行实例列表
instanceNum	int	旅行实例（旅客）个数
cityNum	int	城市总个数
sysTime	Time	系统时间，单位为小时
logFile	std::ofstream*	日志文件输出流
vehKindStr	std::vector<std::string>	交通工具类型字符串
dangerStr	std::vector<std::string>	城市风险类型字符串

3. 类方法

方法名	返回值类型	参数类型	说明
私有			
checkProcessExist	无	无	判断是否已有此程序实例运行
startLogSystem	无	无	启动日志记录系统
startSysClock	无	无	启动系统时钟子线程
incSysTime	无	无	刷新系统时钟
initCityList	无	无	初始化城市列表，存储到 cityList
initSchedule	无	无	初始化时刻表，存储到 schedule
startStatusRefresher	无	无	启动系统刷新器子线程
updateSystem	无	无	刷新旅客状态
公有			
Control	无	无	初始化系统核心,分配资源
~Control	无	无	关闭系统，释放资源
addTravelInstance	bool	InputData&	增加旅行实例，成功返回 true，失败返回 false
IDtoCity	City*	CityID	将城市 ID 转换为城市指针
getWaitTime	Time	Time，Time	根据两个交通时间间隔返回等待时间
pause	无	无	暂停系统时钟
proceed	无	无	继续系统时钟
isPaused	bool	无	判断系统是否处于暂停状态，是返回 true，否则返回 false
openSupport	无	无	打开在线支持页面

2.1.2 城市类 City（DataStructure.h/.cpp）

1. **职责：**存储城市信息，计算停留风险值

2. **类属性**

属性名	数据类型	说明
私有		
ID	CityID	城市唯一标识符，从 0 开始递增分配给每个城市
name	CityName	城市名称

kind	CityKind	城市风险类型
pos	QPointF	城市在地图中的坐标

3. 类方法

方法名	返回值类型	参数类型	说明
公有			
getIconPos	QPointF	无	获取图标绘制位置
getLinePos	QPointF	无	获取路线绘制位置
getCityDangerIndex	DangerIndex	Time	根据停留时间计算风险值
toString	std::string	无	返回城市信息（用于调试）

2.1.3 交通工具类 Vehicle（DataStructure.h/.cpp）

1. 职责：存储交通工具信息，计算交通工具风险值，提供 UI 输出接口

2. 类属性

属性名	数据类型	说明
私有		
ID	VehID	交通工具唯一标识符，从 0 开始递增分配给每个交通工具
name	VehName	交通工具名称
kind	VehKind	交通工具类型
srcCity	City*	始发城市指针
dstCity	City*	终到城市指针
startTime	Time	交通工具开点，当日 24 小时制时间
runTime	Time	交通工具运行时间，单位为小时

3. 类方法

方法名	返回值类型	参数类型	说明
公有			
getArriveTime	Time	无	获取交通工具到点，当日 24 小时制时间
getVehDangerIndex	DangerIndex	无	获取该交通工具风险值（唯一确定）
toUIString	std::string	无	车次相关信息（用于 UI 展示）

2.1.4 旅行实例（旅客）类 TravelInstance（DataStructure.h/.cpp）

1. 职责：存储单个旅行实例信息，计算旅客实时位置

2. 类属性

属性名	数据类型	说明
私有		
ID	TravelID	旅客唯一标识符，从 0 开始递增分配给每个旅客
kind	PlanKind	旅行类型，分为“限时”和“非限时”
beginCity	City*	旅行起点城市指针
endCity	City*	旅行终点城市指针
limitTime	Time	旅行限时（仅对于限时旅行）
beginTime	Time	旅行请求发出时间
lastTime	Time	旅行预计持续时间
plan	Plan	旅行计划
dangerIndex	DangerIndex	旅行综合风险值
keyTime	std::vector<Time>	旅行关键时间节点

3. 类方法

方法名	返回值类型	参数类型	说明
公有			
TravelInstance	无	（各属性）	构建旅行实例，完善对象属性，尤其是生成 keyTime
getStatusUI	std::string	无	UI 展示返回旅客实时位置信息
getStatusLog	std::string	无	日志输出旅客实时位置信息
toStringLog	std::string	无	日志输出旅行信息

2.1.5 交互层输入封装类 InputData（DataStructure.h/.cpp）

1. 职责：封装用户输入，递交控制层

2. 类属性

属性名	数据类型	说明
公有		
beginID	CityID	用户输入的旅行起点城市唯一标识符（ID）
endID	CityID	用户输入的旅行起点城市唯一标识符（ID）
request	Request	用户输入的旅行请求类型，分为“限时”和“非限时”
beginTime	Time	旅行开始时间，可以从系统获取，也可用户自行输入
limitTime	Time	用户输入的旅行限时（仅对于限时旅行）

3. 类方法：此类实例化后仅为数据对象，无主要的类方法

2.2 算法层

2.2.1 旅行计划生成器类 PlanGenerator（GeneratePlan.h/.cpp）

1. 职责：是系统最重要的模块之一，负责根据旅行请求生成最低风险旅行方案

2. 类属性

属性名	数据类型	说明
私有		
verList	verNode*	算法关键数据结构——邻接表的结点列表
nodeNum	int	算法关键数据结构——邻接表的结点数目
nodeID	int**	算法关键数据结构——邻接表的结点编号

3. 内嵌类

类名	类说明	属性名	属性类型	属性说明
私有				
arcNode	邻接表的边结点	verID	int	边的后继节点编号
		dangerIndex	DangerIndex	边的权值
		nextArc	arcNode*	与该边同前驱的下一条边
verNode	邻接表的顶点结点	verID	int	顶点结点编号
		firstArc	arcNode*	以该顶点为前驱的第一条边
tmpNode	算法所需顶点结点类型	verID	int	顶点结点编号
		dangerIndex	DangerIndex	边的权值

4. 类方法

方法名	返回值类型	参数类型	说明
公有			
PlanGenerator	无	无	构建算法所需的关键数据结构
~PlanGenerator	无	无	释放算法所需的数据结构
generatePlan	无	（旅行请求）	算法核心，根据要求生成旅行方案
node2CityID	CityID	int	结点编号转换为城市 ID
node2Time	Time	int	结点编号转换为时间

2.3 交互层

【注：交互层模块因其特点，并无重要的类属性，因此只列举类方法】

2.3.1 主窗体类 COVID_19_Travel_System（COVID_19_Travel_System.h/.cpp）

1. **职责：**程序主界面，提供信息交互

2. **类方法**

方法名	说明
公有	
COVID_19_Travel_System	主窗体类的构造函数，用于初始化界面
callMainUIProceed	其它子窗体发出时钟继续命令的接口
getCurInsIndex	获取 UI 当前选择的旅旅客编号
getUI	供其它窗口访问主窗口元素
私有	
startUIRefresher	启动界面刷新器子线程
refreshUITime	刷新界面时间显示
槽函数	
refreshUI	刷新界面旅客详情
refreshInfo	刷新全部界面内容
UIPause	交互层向控制层发出暂停时钟请求
UIProceed	交互层向控制层发出继续时钟请求
setLimitStatus	控制“限时”输入框可用性
onPressSubmit	响应“提交旅行信息”按钮操作
onPressViewDetails	响应“查看详细信息”按钮操作
onPressVehicleQuery	响应“航班/列车/汽车信息查询”按钮操作
onPressCityDangerQuery	响应“城市风险等级查询”按钮操作
信号	
sendRefreshSignal	定时发出刷新界面的信号

2.3.2 实时地图模块 Map（Map.h/.cpp）

1. **职责：实时反映旅客位置和状态信息**

2. **类属性**

属性名	数据类型	说明
私有		
rePaintSignal	QTimer*	地图刷新定时器
curInsStatus	Status	当前旅行实例的状态

3. **类方法**

方法名	说明
公有	

Map	初始化地图模块
私有	
paintEvent	绘图事件
getIconKind	根据旅行状态选择对应图标
getIconPos	获取图标绘制位置
toPreciseTime	常规精度时间转换到高精度时间
getIconDeltaPos	计算下一时刻图标位移
槽函数	
refreshProgressBar	刷新旅行进度条

2.3.3 其它子窗体类

【注：子窗体类一般用于静态信息展示，因此类属性和类方法均不详细列出】

类名	所在文件名	说明
PlanViewer	PlanViewer.h/.cpp	旅行方案查询结果展示窗口
VehicleQuery	VehicleQuery.h/.cpp	交通工具查询窗口（实用工具，和课设要求无关）
CityDangerQuery	CityDangerQuery.h/.cpp	城市风险等级查询窗口（实用工具，和课设要求无关）

3 关键数据结构

3.1 控制层

3.1.1 Control 类中的数据结构

数据结构名称	数据结构类型	实现方法	内容
城市列表	线性表	vector 容器类	城市对象指针
时刻表	线性表	vector 容器类	交通工具对象指针
旅行实例（旅客）列表	线性表	vector 容器类	旅行实例（旅客）对象指针

3.1.2 TravelInstance 类中的数据结构

数据结构名称	数据结构类型	实现方法	内容
旅行计划	线性表	vector 容器类	旅客需要乘坐的交通工具序列（按时间排序）
关键时间节点	线性表	vector 容器类	旅客每次上车，以及每到达一个城市的时刻序列，刻画了旅行实例的时间特性

3.2 算法层

3.2.1 风险图

1. 数据结构定义

设城市数目为 M，理论最大旅行持续时间为 N（单位：小时），建立有向图 $G=(V,E)$，其中结点数为 $M\times N$，即为每个城市的每个时刻定义一个结点，并设 `V[i][j]` 表示城市 i 在时刻 j 所代表的结点。

对于任意城市 A，若其单位风险值为 W，对于任意的 $0\leqslant i<N$，有一条以结点 `V[A][i]` 为起点，以结点 `V[A][i+1]` 为终点的边，且权值为 W。

对于任一交通工具，若其于当日 t 时刻在城市 A 出发，运行 l 小时后到达城市 B，乘坐风险为 W，则对于任意的 $0\leqslant j<N/24$，有一条以结点 `V[A][t+24*j]` 为起点，以结点 `V[B][t+l+24*j]` 为终点的边，且权值为 W。

2. 数据结构类型：有向图

3. 实现方法：邻接表

4. 代码示例：

```
class arcNode {
public:
    int verID;                      // 终点顶点编号
    DangerIndex dangerIndex;        // 边权值，此处为风险值
    arcNode* nextArc;               // 与该边具有相同起点顶点的下一条边
    arcNode() = delete;
    arcNode(int verID_, DangerIndex dangerIndex_)
        : verID(verID_), dangerIndex(dangerIndex_), nextArc(nullptr) {
    }
};
class verNode {
public:
    int verID;                      // 顶点编号
    arcNode* firstArc;              // 以该顶点为起点的第一条边
};
```

3.2.2 结点索引数组

1. 数据结构定义

本数据结构服务于上文提到的有向图，为结点提供城市-时刻索引，每个城市的每个时

刻定义一个结点，并设 V[i][j]表示城市 i 在时刻 j 所代表的结点序号。

2. **数据结构类型：**矩阵

3. **实现方法：**二维数组

3.2.3 最短路径算法执行过程中的结点堆

1. **数据结构定义：**在算法执行过程中可以方便的寻找当前最小的距离作为观察点

2. **数据结构类型：**小根堆

3. **实现方法：**优先队列（priority_queue 容器类），基于二叉堆

3.3 交互层

【注：交互层因其特点，无主要的数据结构】