# 数据结构说明和数据字典

### Passenger类

class **Passenger**

{

public:

**Passenger**() { m\_status = -1; m\_time = 0; };

//默认构造函数

void **initial\_a\_Passenger**(int day, int leave\_time, int source, int destination, int strategy, int passenger\_num, int max\_time);

//初始化限时模型下的旅客

void **initial\_a\_Passenger**(int day, int leave\_time, int source, int destination, int strategy, int passenger\_num);

//初始化非限时模型下的旅客

~**Passenger**() {};

//析构函数

void **travel\_simulation\_DMS**(int trans, double danger, int temp\_time, Passenger& A);

//非限时模型下的旅行模拟，用来求出最优旅行方案

void **travel\_simulation\_PDMS**(int trans, double danger, int temp\_time, int elapsed\_time, Passenger& A);

//限时模型下的旅行模拟，用来求出最优旅行方案

void **initial\_min\_danger\_trans\_and\_stack**();

//初始化旅行模拟辅助数组

void **dbg\_output\_min\_danger\_trans**();

//测试用输出最小风险

void **dbg\_get\_source**();

//测试用获取目的地

void **stack\_back\_all**();

void **stack\_back\_1**(int trans);

void **stack\_back\_2**(int trans);

//退栈函数，用来辅助旅行模拟函数

void **get\_plan**();

//用于获取模拟的计划

int **label\_pos\_x**(int i);

//获取label横坐标

int **label\_pos\_y**(int i);

//获取label纵坐标

QString **travel**();

//旅行函数 按照随时间进行旅客移动 返回QString类型以便文字显示

QString **inquire\_now**();

//查询当前状态函数

QString **inquire\_plan**();

//查询计划函数

QLabel \*my\_label;

//图标

int **get\_status**() { return m\_status; };

//获知旅客状态

private:

int m\_leave\_day;

//旅客出发的日期

int m\_leave\_time;

//旅客出发的时间

int m\_source;

//出发地

int m\_stay\_city\_or\_transportation;

//当前状态,置0为等待状态，置其他为在旅行中的时间

int m\_destination;

//目的地

int m\_strategy;

//出行策略

int m\_max\_time;

//限时情况下的乘客最大时间

int m\_time;

//记录旅客总耗时

int m\_how\_travel\_city[MAX\_TRANS + 1][4];

//二维数组

// 0 1 2 3

// 城市代号 交通工具代号 第几列车次 是否进行了本次的旅途

int m\_passenger\_num;

//旅客的序号

int m\_status;

//旅行状态,0代表旅行中，1代表旅行已经结束,-1代表未创建,2代表无法出行

int m\_stack[MAX\_TRANS + 1][2];

/\*一个栈 用来为该旅客模拟出行情况（用于算法中的模拟）

二维数组 0 1

城市 车次

\*/

int m\_min\_danger\_trans[MAX\_TRANS + 1][2];

//暂存最小风险策略的二维数组（用于算法中的存储）

//二维数组情况同上

double m\_min\_danger;

//记录最小风险

};

### 2.my\_set.h中的全局变量

extern const char BEIJING[10] = "BEIJING";

extern const char SHANGHAI[10] = "SHANGHAI";

extern const char GUANGZHOU[10] = "GUANGZHOU";

extern const char HANGZHOU[10] = "HANGZHOU";

extern const char SHENYANG[10] = "SHENYANG";

extern const char HAERBIN[10] = "HAERBIN";

extern const char CHENGDU[10] = "CHENGDU";

extern const char NANJING[10] = "NANJING";

extern const char XIAN[10] = "XIAN";

extern const char WUHAN[10] = "WUHAN";

extern const char MY\_ERROR[10] = "ERROR!";

//城市字符

extern const char FLIGHT[10] = "FLIGHT";

extern const char TRAIN[10] = "TRAIN";

extern const char CAR[10] = "CAR";

//交通工具字符

#define MAX\_TRANS 5

//定义最大转车次数，方便回溯法剪枝

#define MAX\_DANGER\_DMS 20

//定义最大可接受的风险大小

#define MAX\_DANGER\_PDMS 20

//定义最大可接受的风险大小

#define MAX\_DANGER\_WITH\_TRANS\_DMS 50

//定义最大可接受的风险大小

#define MAX\_DANGER\_WITH\_TRANS\_PDMS 50

//定义最大可接受的风险大小

//以上参数为回溯法剪枝所使用的参数

#define DANGER\_MIN\_STRATEGY 0

//风险最小策略

#define PUNCTUAL\_DANGER\_MIN\_STRATEGY 1

//规定时间的风险最小策略

#define DANGER\_MIN\_WITH\_TRANS\_STRATEGY 2

//风险最小策略

#define PUNCTUAL\_DANGER\_MIN\_WITH\_TRANS\_STRATEGY 3

//规定时间的风险最小且包含交通工具策略

#define SOURCE 0

#define DESTINATION 1

#define TRANSPORTATION 2

#define START\_TIME 3

#define FINISH\_TIME 4

#define ALL\_TIME 5

//timetable数组中的每列所对应的条目

#define MAX\_PASSENGER 50

//最大接待乘客数

double city\_table[10][2];

/\*城市情况表

0 1

城市 风险

\*/

double trans\_table[3];

//交通工具风险情况

int timetable[10][20][6];

/\*

第一列代表城市 每个城市对应一个二维数组

二维数组

0 1 2 3 4 5

始发地 目的地 交通工具 出发时间 到达时间 消耗的总时间

该二维数组按照出发时间升序排列

第二维设定为20的原因是：经查表记录最多班次的城市北京所拥有的发车次数为16 故设定20为最大值

\*/

int now\_year = 2020;

int now\_month;

int now\_day;

int now\_hour;

//时间

### 3.mainwindow.cpp中的变量

static int my\_passenger = 0;

//乘客数量

static int lock = 0;

//控制程序

static int is\_start = 0;

//是否开始

static int is\_pause = 0;

//是否暂停

static QString log\_time;

//供日志使用的字符串形式的时间

static Passenger P[MAX\_PASSENGER];

//一个乘客的数组