旅行模拟系统——数据结构说明和数据字典

Travel_System

一、数据结构

在此模块给出存储的示例以及基本数据结构信息，完成的数据结构信息和相应变量名将在数据字典模块给出完整介绍。

（一）旅客信息

旅客信息使用链表进行存储。

存储示例：

```
Current_ptr->Depart_Place = Depart;
Current_ptr->Dest_Place = Dest;
Current_ptr->Seq = Seq;
Current_ptr->Time_Req = Time_Req;
Current_ptr->Risk_Req = Risk_Req;
Current_ptr->Price_Req = Price_Req;
```

（二）地图信息

地图信息采用邻接链表的方式进行存储。

在最开始的设计中，我的地图是写死在程序里的，后面考虑到实际设计中不可能将地图信息完全写死在程序里，这样会大大降低程序的交互性，在有更改地图信息的需求下应该有更加方便的修改模式，所以后期改为：每次启动程序进行地图初始化的时候从外部读入地图信息进行存储

存储示例：（图为外部存储的地图信息）

（三）时刻表信息

时刻表信息并没有利用数据结构进行存储，在计算时间的函数中对时刻表有所体现。大致时刻表如下：

交通工具类型	每天出发时刻
飞机	7
火车	12
汽车	8/12/16/20

其中每天共有 28 列次火车，10 班次飞机，112 车次长途汽车。

（四）其他数据信息

其他数据信息主要有风险值的设定，价格的设定等。
存储示例：

```
const float City_High_Risk = 0.9;
const float City_Medium_Risk = 0.5;
const float City_Low_Risk = 0.2;
#define UNPASSED_CITY 0
#define PASSED_CITY 1
#define START_CITY 2
#define END_CITY 3
#define Airtime_Price 600
#define Traintime_Price 150
#define Bustime_Price 25
#define Bus_Risk 2
#define Train_Risk 5
#define Air_Risk 9
```

二、数据字典

（一）旅客信息

旅客信息存储链表截图：

```
struct Passenger {
    int Seq;
    int Time_Req;
    float Risk_Req;
    int Price_Req;
    QString Depart_Place;
    QString Dest_Place;
    struct Trav_Plan {
        QString Plan_City[20];
        int Plan_Vehicle[20];
        int Plan_Time[20];
        float Plan_risk;
        int Plan_Price;
    }Plan;
    struct Passenger* next;
    struct Passenger* prev;
};
typedef struct Passenger Pas_List;
```

变量名	变量类型	变量功能描述
Seq	int	记录旅客编号
Time_Req	int	记录旅客时间要求
Risk_Req	float	记录旅客风险要求
Price_Req	int	记录旅客价格要求
Depart_Place	QString	记录旅客出发地
Dest_Place	QString	记录旅客目的地
Plan_City	QString	记录旅客所有经过的城市（包括出发地和目的地）
Plan_Vehicle	int	记录旅客所有使用过的交通工具类型
Plan_Time	int	记录旅客在每个状态（停留在某城市的时间区间，停留在某交通工具上的时间区间）的时间。
Plan_Risk	float	记录旅客旅行策略风险
Plan_Price	int	记录旅客旅行策略花费时间

（二）地图信息

```
typedef struct City C;
typedef struct Map M;
struct City {
    QString City_Name;
    float City_Risk;
    int City_State;
    int Count_Adj_City;
    struct Record {
        QString Adj_Name;
        int Seq_Name;
        int Mark_Air;
        int Mark_Train;
        int Mark_BUS;
        int Air_Time;
        int Train_Time;
        int Bus_Time;
    }Record_Adj_City[12];
    C* Adj_City[12];
    C* next;
};

struct Map {
    C* Map_City[12];
};
```

变量名	变量类型	变量功能描述
City_Name	QString	城市名
City_Risk	float	城市风险值
City_State	int	城市状态（分为经过，未经过，起始，终点四种状态，分为旅行策略路线时使用）
Count_Adj_City	int	邻接（有交通工具可直达）城市数目
Adj_Name	int	邻接城市名
Seq_Name	int	邻接城市序号
Mark_Air	int	与邻接城市是否有飞机直达
Mark_Train	int	与邻接城市是否有火车直达
Mark_BUS	int	与邻接城市是否有汽车直达
Air_Time	int	两城市飞机飞行时间，若无飞机直达则为-1
Train_Time	int	两城市火车单程时间，若无火车直达则为-1
Bus_Time	Int	两城市汽车单程时间，若无汽车直达则为-1
Adj_City	City	用于将邻接信息赋给地图

Map_City	City	记录地图信息

（三）其他数据

```
const float City_High_Risk = 0.9;
const float City_Medium_Risk = 0.5;
const float City_Low_Risk = 0.2;
#define UNPASSED_CITY 0
#define PASSED_CITY 1
#define START_CITY 2
#define END_CITY 3
#define Airtime_Price 600
#define Traintime_Price 150
#define Bustime_Price 25
#define Bus_Risk 2
#define Train_Risk 5
#define Air_Risk 9
int count_record_num[20];
int planprice[20];
float planrisk[20];
int planvehi[20][8];
int plantime[20][16];
int a = 0, b = 0;
    QString record[20][12];
```

变量名	变量类型	变量功能描述
City_High_Risk	float	高风险城市的风险值
City_Medium_Risk	float	中风险城市的风险值
City_Low_Risk	float	低风险城市的风险值
UNPASSED_CITY	无	未经过的城市
PASSED_CITY	无	已经过的城市
END_CITY	无	终点城市
START_CITY	无	出发地城市
Airtime_Price	无	每时间单位飞机票价钱
Traintime_Price	无	每时间单位火车票价钱
Bustime_Price	无	每时间单位汽车票价钱
Bus_Risk	无	每时间单位汽车风险值
Train_Risk	无	每时间单位火车风险值
Air_Risk	无	每时间单位飞机风险值
Count_record_num	int	计算一个旅客所有可能的线路的数目
planprice	int	旅客的旅行计划价格
planrisk	float	旅客的旅行计划风险值
planvehi	int	旅客的旅行计划载具(每一条可能的线路的时间区域记录)
plantime	int	旅客的旅行计划时间(每一条可能的线路的使用交通工具记录)
a	int	用于筛选旅行线路
b	int	用于筛选旅行线路

record	QString	记录一个旅客所有可能的旅行线路

```
int Num_Pas = -1;
int Timer_Time = 0;
Pas_List* Pas_ptr = NULL, * Current_ptr = NULL, * Last_ptr = NULL;
```

变量名	变量类型	变量功能描述
Num_Pas	int	用于分配旅客序号
Timer_Time	int	用于显示时间
Pas_ptr	Pas_List	旅客链表头节点
Current_ptr	Pas_List	旅客链表操作节点
Last_ptr	Pas_List	旅客链表尾节点

```
QTimer *ptimer;
QTime baseTime;
QString showStr;
```

变量名	变量类型	变量功能描述
ptimer	QTimer	定时器对象指针
basetime	QTime	基准时间
showStr	QString	用于显示的字符串

（四）主要函数

```
void paint_hashen();
void paint_shenbei();
void paint_beinan();
void paint_beiwu();
void paint_beixi();
void paint_lanxi();
void paint_shenlan();
void paint_xiwu();
void paint_nanwu();
void paint_xicheng();
void paint_xikun();
void paint_chengkun();
void paint_chengchang();
void paint_wuchang();
void paint_wufu();
void paint_changkun();
void paint_changguang();
void paint_fuguang();
void paint_chengfu();
void paint_nanguang();
```

变量名	变量类型	变量功能描述

paint_XXXX	void	控制在图形界面显示

```
void on_AddPas_clicked();
void on_Start_clicked();
void on_Pause_clicked();
void on_Stop_clicked();
void Update_Num();
void on_Search_clicked();
```

函数名	函数类型	函数功能描述
on_AddPas_clicked	void	点击增添按钮执行函数
on_Start_clicked	void	点击开始/继续按钮执行函数
on_Pause_clicked	void	点击暂停按钮执行函数
on_Stop_clicked	void	点击结束按钮执行函数
Update_Num	void	定时器每次超时执行函数
on_Search_clicked	void	点击查询按钮执行函数

```
QString Search_Pas_State(int Seq, int Time){ ...}
void PrintAdd(int j, int m) { ...}
int Assign_To_Pas(int i) { ...}
void Add_Pas(int Seq, int Time_Req, int Risk_Req, int Price_Req, QString Depart, QString Dest){ ...}
void Free_Pas(Pas_List* hPtr){ ...}
```

函数名	函数类型	函数功能描述
Search_Pas_State	QString	查询对应序号和时刻的旅客状态信息
PrintAdd	void	打印旅客信息（用于测试）
Assign_To_Pas	int	用于将分配好的路线赋值给旅客链表
Add_Pas	void	用于新增一名旅客
Free_Pas	void	释放旅客链表

```
float Get_Risk(QString city) { ...}
int Get_Vehi_Type(QString start, QString end) { ...}
int Get_Vehi_Time(QString start, QString end, int type) { ...}
float Get_City_Risk(QString city) { ...}
void Get_Pas_Plan(int init_time) { ...}
void Out_Record(void) { ...}
void Record_All_Path(C* Start, C* End) {//only record path's length <=5 ...}
C* Match_Start(M* Map_China, C* Start, QString name) { ...}
C* Match_End(M* Map_China, C* End, QString name) { ...}
void Get_All_Path(C* Depart, C* Dest, int City_Num, M* map) { ...}
int Judge_Req_avail(QString s, QString e) { ...}
void Find_All_Path(QString s, QString e) { ...}
```

函数名	函数类型	函数功能描述
Get_Vehi_Type	int	获取两城市之间的交通工具类型
Get_Vehi_Time	int	获取两城市间对应交通工具的单程时间
Get_City_Risk	float	获取对应城市风险值
Get_Pas_Plan	void	函数具备时刻表功能，可获取精确到时间区间的旅客计划
Record_All_Path	void	筛选出途径城市数<=5 的旅行策略
Match_Start	City	匹配出发地
Match_End	City	匹配目的地
Get_All_Path	void	获取全部从出发地到目的地的可能旅行策略路径
Find_All_Path	void	在获取路径前完成初始化等工作