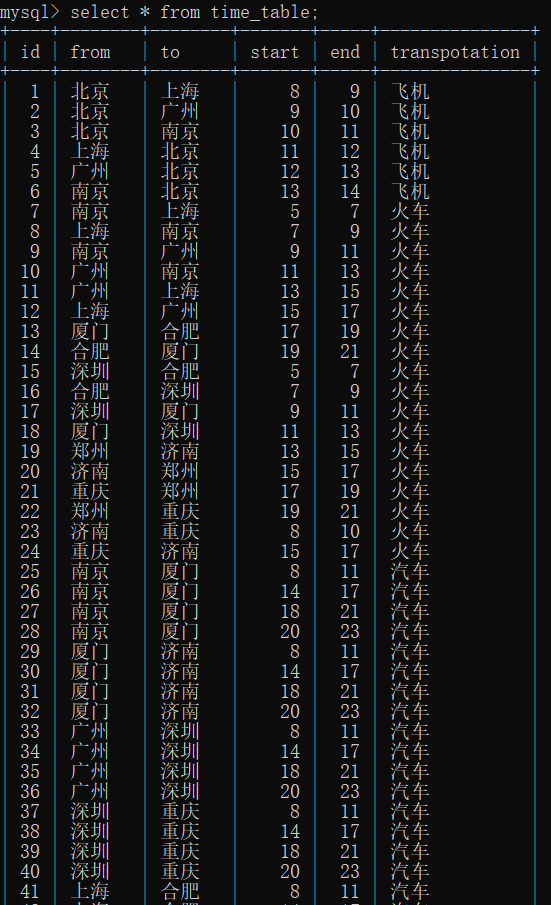
# 数据结构说明

### 数据库数据说明：

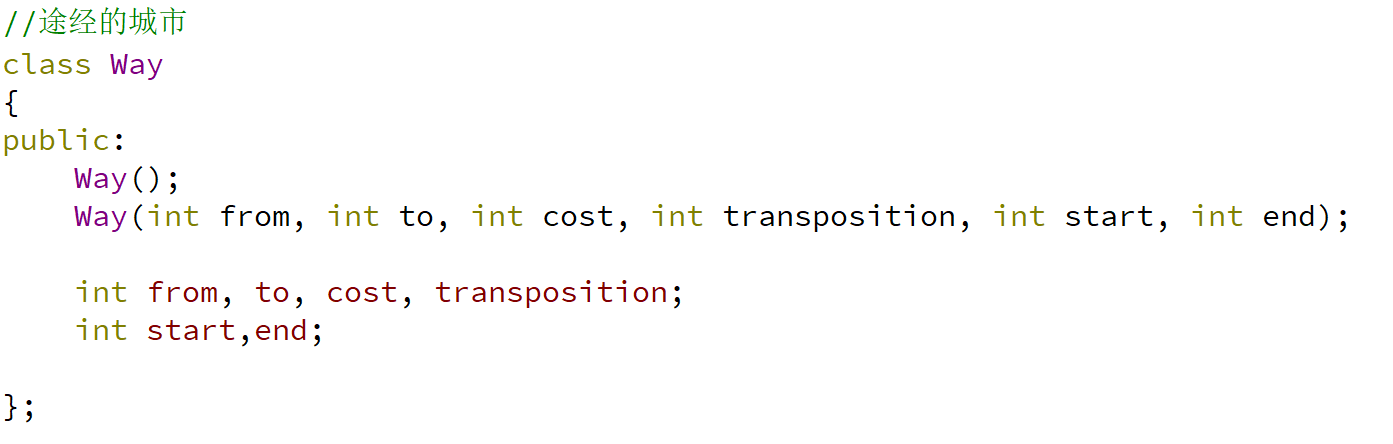
city中存储各城市风险值

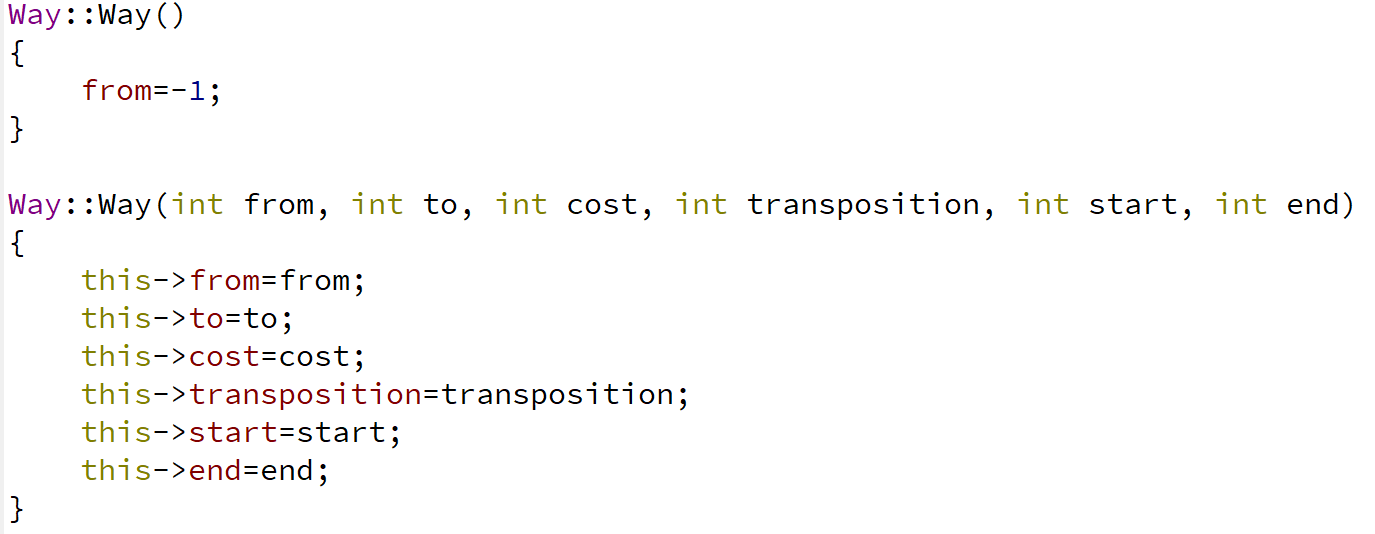


time\_table中存储时刻表



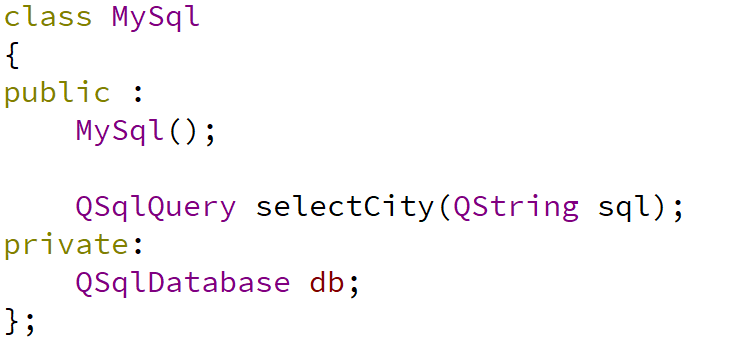
**Way模块：**

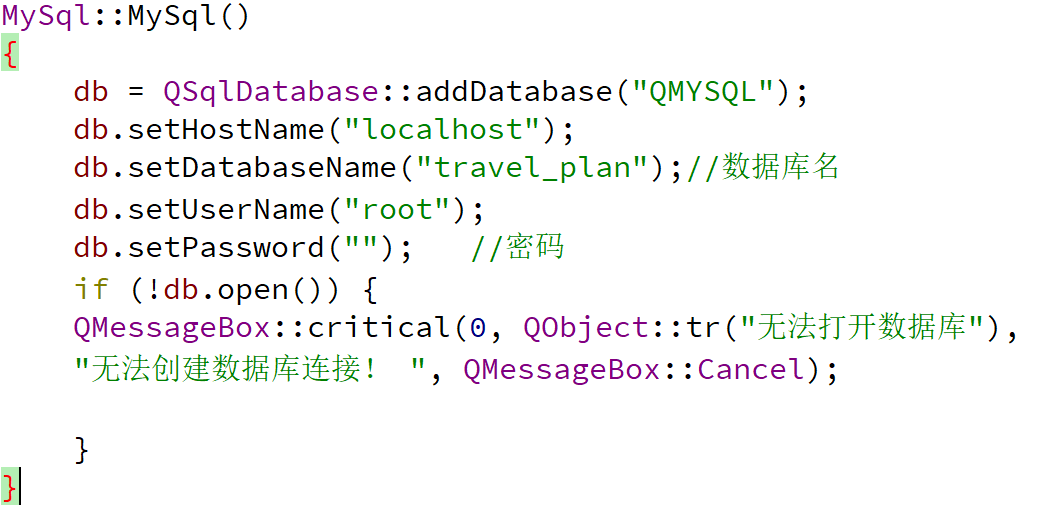


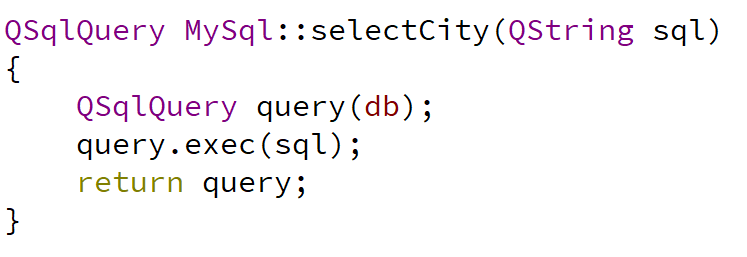


Way用于存放路径的时刻表，包含起止城市、起止时间、交通工具、耗时

**MySQL模块：**







MySql初始化时，会连接到数据库，selectCity可通过输入的sql语句，在时刻表中查询符合要求的数据，并返回QSqlQuery类型

**Map模块：**

数据结构

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 标识符 | 类型 | 详细描述 |
| 交通工具的类型 | type | int | 用来描述需要绘图的交通工具的类型 |
| 开始城市 | from | int | 存储绘图时开始城市位置 |
| 终止城市 | to | int | 存储绘图时终止城市位置 |
| 现在位置 | currentPosition | int [2] | 存储绘图时现在位置 |
| 目的位置 | destination | int [2] | 存储绘图时目的位置 |
| 城市位置 | city | int [10][2] | 存储所有城市位置 |
| x方向单位移动距离 | xMove | int | 存储绘图时每次计时器超时x方向单位移动距离 |
| y方向单位移动距离 | yMove | int | 存储绘图时每次计时器超时y方向单位移动距离 |
| 地图模块计时器 | mTimer | QTimer | 地图模块计时器 |
| 飞机移动 | plane\_move | void | 飞机移动函数 |
| 火车移动 | train\_move | void | 火车移动函数 |
| 汽车移动 | car\_move | void | 汽车移动函数 |
| 自定义析构函数 | end | void | 自定义析构函数 |
| 计时器超时 | mTimer\_timeout | slot | 计时器超时 |
| 获取数据 | getData | slot | 从主窗口获取数据 |
| 显示旅客 | showP | slot | 显示旅客所在位置 |

**总体功能：**

当getData函数从主窗口获得交通工具、起始城市、终止城市时，根据交通工具的速度和城市之间的距离，计算出每次移动在x，y方向上所需移动的距离，计时器每1s超时一次，也就是程序中的6分钟。在开始时，旅客在起始城市显示，根据交通工具的类型选择移动函数，在此期间，旅客消失，交通工具显示，当到达目的城市时，交通工具消失，旅客显示。

**Widget模块：**

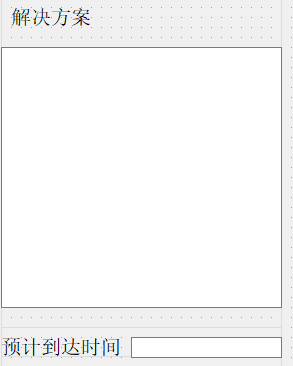
**输入：**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识符 | 类型 | 介质 | 描述 |
| 策略 | strategy | int | 选择框 | 储存用户选择的策略 |
| 起始城市 | fromCity | int | 选择框 | 储存用户选择的出发城市 |
| 终止城市 | toCity | int | 选择框 | 储存用户选择的目的地 |
| 时间限制 | timeLimite | int | 文本输入框 | 储存用户输入发时间限制 |

**输出：**





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识符 | 类型 | 介质 | 描述 |
| 路径输出 | plan | Way | 文本框 | 输出旅客路径 |
| 预计到达时间 | arriveTime | int [3] | 文本框 | 输出旅客预计到达时间 |
| 现在时间 | now | int [3] | 文本框 | 输出现在时间 |

**数据结构：**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 标识符 | 类型 | 详细描述 |
| 数据库 | mysql | MySql | 初始化存储时刻表的数据库 |
| 现在时间-天 | day | int | 现在时间-天 |
| 现在时间-小时 | hour | int | 现在时间-小时 |
| 现在时间-分钟 | minute | int | 现在时间-分钟 |
| 时间输出文本 | txtContext | QString | 以文本形式输出现在时间 |
| 策略 | strategy | int | 输入的策略 |
| 起始城市 | from | int | 输入的起始城市 |
| 终止城市 | to | int | 输入的终止城市 |
| 时间限制 | timeLimite | int | 输入的时间限制 |
| 是否按下按钮 | pushBotton | bool | 是否按下按钮 |
| 定时器 | fTimer | QTimer | 主窗口定时器 |
| 子窗口 | mapWidget | Widget | 子窗口 |
| 交通工具的类型 | type | int | 绘图传递的交通工具的类型 |
| 现在起始城市 | currentFrom | int | 现在起始城市 |
| 现在目的城市 | currentTo | int | 现在目的城市 |
| 每个城市的移动策略 | plan | Way [10] | Dijkstra算法所需的每个城市的移动策略 |
| 每个城市的到达时间 | arriveTime | int [10][3] | Dijkstra算法所需的每个城市的到达时间 |
| 是否成功找到路径 | success | bool | 是否成功找到路径 |
| 暂时存储路径 | tempPlan | Way [10] | 暂时存储路径，用来更新plan |
| 旅客路径 | path | QStack<int> | 旅客路径 |
| 画图路径 | mapPath | QStack<int> | 画图所需的路径 |
| 当前最小风险 | minCost | double | DFS算法当前最小风险 |
| 所耗时间 | costTime | int [2] | 单条路径所耗时间 |
| 风险值 | cost | double | 单条路径所需风险值 |
| 城市风险值 | value | double [10] | 存储每个城市的风险值 |
| 现在时间 | currentTime | int [3] | 现在时间 |
| 是否经过城市 | throughCity | bool [10] | 标识城市是否经过 |
| 城市距初始城市距离 | distance | double [10] | 城市距初始城市的风险值 |
| 制定路径 | Makeplan | void | 根据plan的内容，寻找路径，并存储在path中 |
| 寻找策略对应的路径 | findWay | void | 根据策略选择不同的算法，同时对数据进行初始化 |
| Dijkstra算法 | Djkstra | void | Dijkstra算法寻找路径 |
| 更新城市 | updateCity | void | Dijkstra算法更新城市 |
| 深度优先搜索算法 | DFS | void | 深度优先搜索算法 |
| 计算总消耗时间 | calculateTotalCost | void | 深度优先搜索算法计算总消耗的时间 |
| 计算风险值 | calculateCost | void | 计算单条路径风险值 |
| 时间更新 | updateCurrentTime | void | 定时器超时时，更新时间 |
| 定时器超时 | fTimer\_timeout | slot | 定时器超时 |
| 按下提交按钮 | on\_submit\_clicked | slot | 按下提交按钮，传入数据 |
| 按下加速按钮 | on\_pushButton\_clicked | slot | 按下加速按钮，时间前进一小时 |
| 传递数据 | sendData | signal | 向子窗口传递数据 |
| 显示旅客 | showPassenger | signal | 子窗口显示旅客 |

功能：获取输入、根据不同策略寻找路径，并向地图子窗口传递绘制所需的信息，实现数据交换和主要功能。