问题描述

10部电梯的仿真运行，仿真过程包含人乘坐电梯的仿真行为

命名规则

总体遵从驼峰式命名，

标识符的命名用英文，避免使用其它语言或汉语拼音，名称大致能表示实际意义。

标识符一般由一个或几个单词构成，单词的首字母大写用于分隔单词。

变量名：

变量用名词或词组。普通变量首字母小写 如totalSimulationTime

常量首字母大写 如FixedWidth

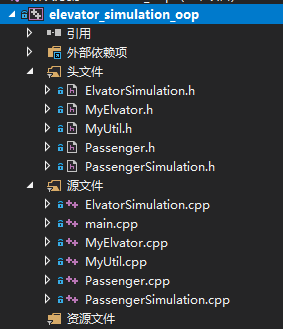
函数名：

函数名用动词或动词短语，如dealCustomerRequest（）

类名：

第一个字母大写。如PassengerSimulation

设计思路



项目目录

共分为五个类：

**ElvatorSimulation MyElvator MyUtil Passenger passengerSimulation类 Passenger 乘客类**

包含乘客的各种属性如下

public:

int tokentimes; //客人要乘坐电梯的次数，文档中的要求乘坐L次，第L+1次回到一层，这个变量包含第L+1次

private:

int waitFloor; //客人等待电梯的楼层

int distFloor; //客人要去往的楼层

int belongingNeedSpace; //客人携带的随身物品占用的空间

int waitElvatorNum; //客人要等的电梯，即接客的电梯

int takingtimes; //客人已经乘坐电梯的次数，

int waitTimeforNext; //完成一次乘梯活动后，休息的时间，即两次乘梯之间的间隔，随机产生

这是基本的model

**passengerSimulation 乘客仿真类**

是一个单例类，程序中只存在一个实体

包含一个指向仿真人群的指针，还在仿真中等待下一次乘梯的乘客列表，这个列表乘客带有状态，即带了多少东西，在哪一层等；仿真结束完成仿真任务的乘客列表

Passenger在这个类里有实例

具体属性如下

public:

int customerNum; //仿真时间内出现的顾客数量

//queue<Passenger> customerRequest; //顾客请求乘坐电梯的请求队列

Passenger\* passengerPtr; //指向人群的指针

list<Passenger> simulatingPassengers; //还在仿真的乘客列表

list<Passenger> simulatedPassengers; //仿真结束的乘客列表

MyUtil\* myUtil;

**MyElvator 电梯类**

包含电梯的基本属性，如下

typedef struct elevatorNode

{

bool linkageFlag; //联动标志, 0-9每个电梯对应自己的标志，电梯两两一组，共五组，false表示没有接客任务

int elevatorCarryMen; //电梯当前人数

int stopFloor; //电梯在没任务时停的楼层

int distFloor; //电梯接客的目的楼层，即客人等待电梯的楼层

int currentFloor; //电梯当前所在的楼层

bool runningDirectionFlag; //电梯当前的运行方向，true为向上，false为向下

int runningTimeforNextFloor; //电梯上次关门后已经运行的时间

int runTimeforNextFloor; //电梯上次关门到下次开门需要运行的时间

bool runFlag; //电梯是否在运行的标志，为真在运行，否则是空闲状态

bool menLeavingFlag; //电梯是否在上下客人的标志

}elevatorElement;

public:

const int defaultCarryNum = 16;

//friend class PassengerSimulation;

MyUtil\* myUtil = MyUtil::getInstance();

public:

elevatorElement elevator; //电梯基本属性

//对应已经乘坐在每部电梯中的顾客, 按楼层顺序插入，

//每到一个新楼层检查表头元素是否下楼，元素来源于waitingCustomer变量

list<Passenger> carryCustomers;

/\*对应正在等待每部电梯的Person集合，

链表中的Person元素的插入位置按人的楼层排序，

链表空则直接去接第一个请求的客户，否则按电梯运行方向的顺序接顾客

由ElvatorSimulation类往里面填充数据

\*/

list<Passenger> waitingPickUpCustomers;

int maxCarryMen; //最大载客量

PassengerSimulation\* ps;

**ElvatorSimulation 电梯仿真类**

模拟电梯仿真的行为，电梯类，乘客仿真类都在这个类里有实例，是主要的类

其属性如下:

private:

//题目中的十部电梯，其实可以手动输入电梯的数量，然后在专门建立一个类写电梯运行规则，

//这样设计更便于扩展，但是题目没有要求，就偷懒吧

MyElvator myElvator[10];

int totalSimulationTime; //总共的仿真时间

//int customerNum; //仿真时间内出现的顾客

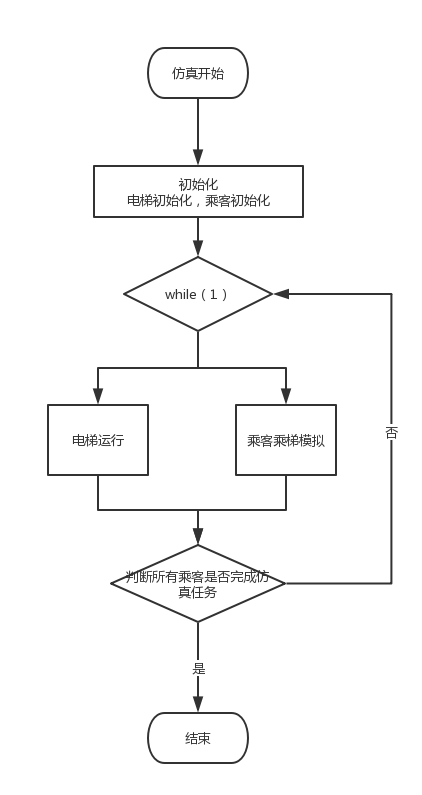
PassengerSimulation\* ps;

**MyUtil 工具类**

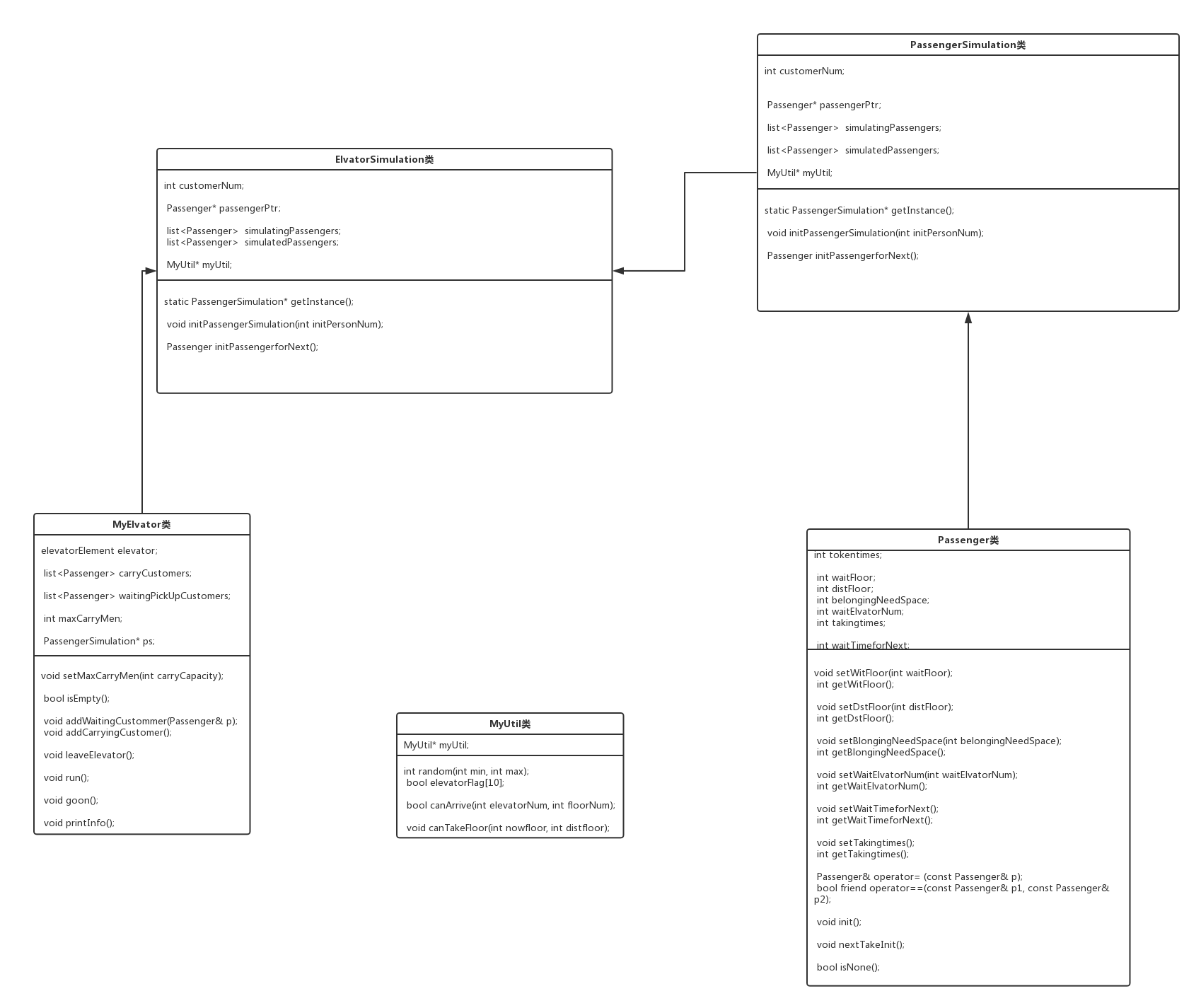
是一个单例类，程序中只存在一个实体

定义电梯的运行规则和随机数生成等一些项目中所需要的工具

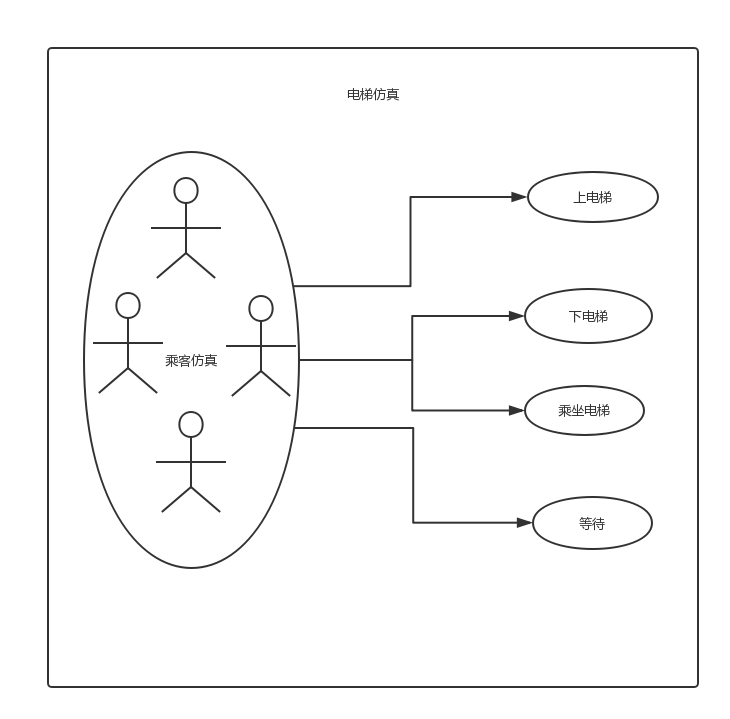
仿真流程图：



UML类图：



UML用例图：



开发进度

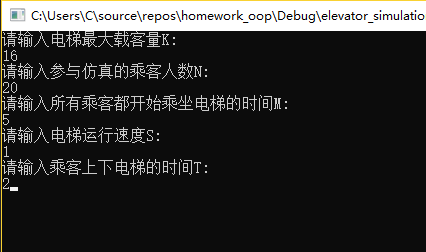
完成了大部分功能

1. 乘客仿真类对每一个乘客进行管理，从开始仿真到结束，模拟乘客乘坐电梯，等待的行为，模拟乘客请求乘坐电梯
2. 电梯仿真类对每部电梯进行管理，电梯初始化，设置电梯对应的参数，如载客量等，然后以秒为单位进行检查电梯是否到下一个楼层，
3. 电梯类实现了电梯自己的动作，检查是否有乘客下，有乘客上，运行到下一层
4. 乘客类，model类，定义乘客的属性，如所在楼层，目标楼层。携带的物品，乘坐的次数
5. 工具类里单独定义了电梯运行规则

电梯运行之后和乘客请求分离开来，乘客请求不会影响电梯的运行

结果展示

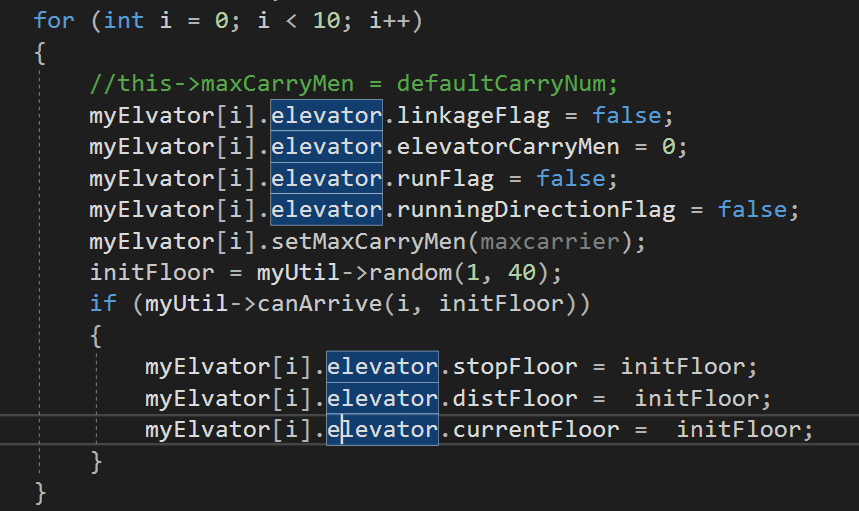
1. 从命令行输入相关参数



2. 仿真开始各电梯随机在其符合运行规则的任意一层

产生随机数后会判断电梯能否到达该层，能才初始化到该层，运行时同理，能到该层才会检查是否需要接人或者有乘客要下：

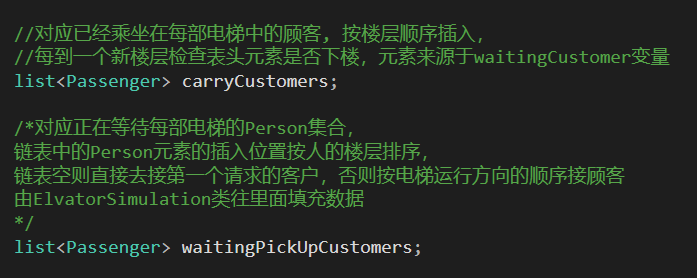
电梯初始化代码：



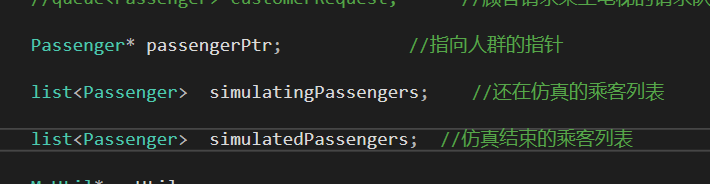
3. 每位乘客初次到达的楼层随机，在合适地方等待电梯

动态生成一批乘客然后初始化后加入仿真列表，每个乘客常坐电梯时从仿真队列取出加入电梯的接客列表，接到后加入乘坐列表，每个乘客的仿真任务完成后加入仿真完成列表

电梯的接客列表和乘坐列表:



乘客仿真的列表：

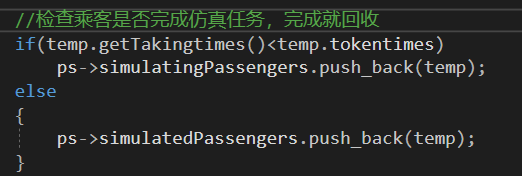


乘客初始化代码：



4. 乘客完成一次后暂停几秒继续，直到所有仿真任务完成，回收

代码：

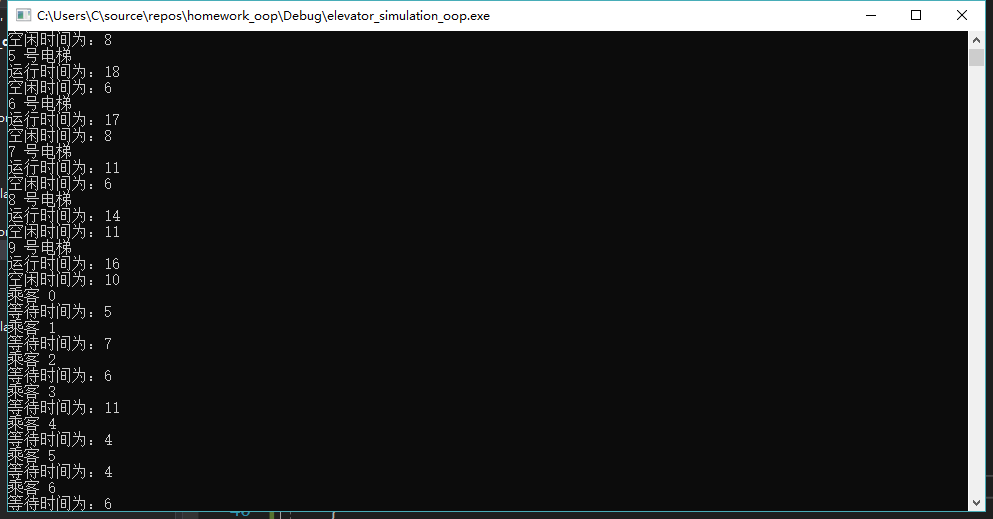


5. 电梯运行由先发出请求者决定

电梯接人的队列按先后顺序依次往后插入，每次先接表头的人如果在接人的运行过程中没有其他顺路的请求

数据结构见3所述

6 . 统计电梯运行时间 和 乘客等待时间

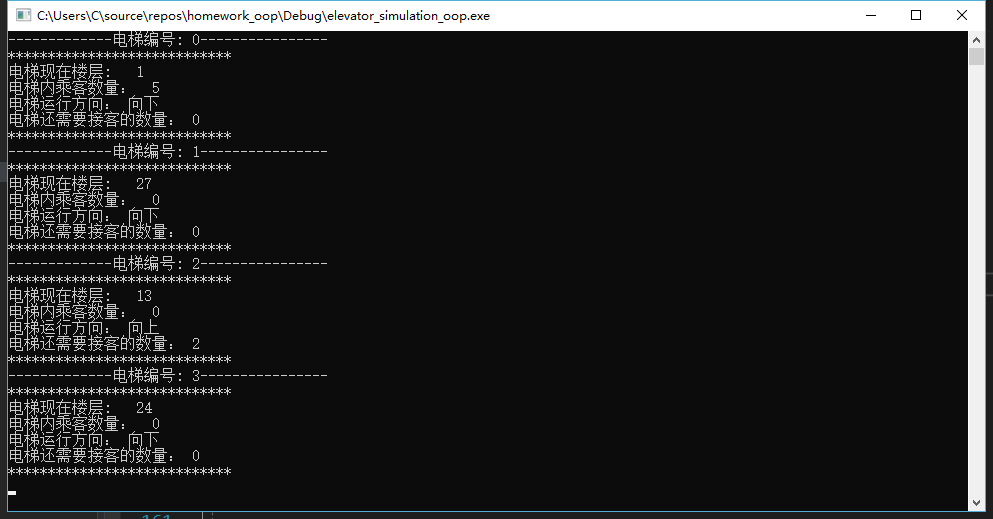


运行截图：

去接人：



0号电梯内已有五人



存在的问题

1. 电梯联动，还存在一些小bug，有时候相互无法通知

2. 没有实时显示顾客的请求