

**Lab Report**

**实验报告**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2024-2025** |
| **Major**: | Software Engineering |
| **Class**: | 2023 |
| **Student Name**: | 吴仲雄 |
| **Student ID:** | 222023321062007 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Queue and Simulation  队列和仿真 | | | |
| Date | | Nov，2024 | Type | | ☑Confirmatory （验证确认型）  ☑Design（设计型）  🗆Comprehensive（综合型） |
| 1. **Objective & Requirements（实验目的）**    1. Understand the concept of container adapter; Know the implementation the queue container adapter in the STL   理解容器适配器的概念；了解STL中队列容器基于容器适配器的实现原理   * 1. Grasp the use of queue container in a real application   掌握队列容器的常用接口和使用方法，能够利用队列容器解决现实应用问题   * 1. Know the concept of simulation and can use simulation to solve a real problem; Know about the queueing theory and the exponential distribution theory   了解仿真的概念和原理，能够使用基于队列的仿真技术解决现实问题；了解排队论理论和指数分布理论，能够利用相关理论知识结合队列编程对现实问题进行理论分析和算法、工程改进。 | | | | | |
| 1. **Experimental environment (**platform and software**)（实验环境）**   Windows 7 (or higher versions) + Visual Studio 2010 (or higher versions) | | | | | |
| 1. Experimental content and design (Main Content, Procedure, Codes and Results)（此部分应包含每一个实验内容的详细设计，含实验思路、详细实验步骤、核心代码说明等）   Task 1  Improve the car wash simulation problem in the following ways based on the codes.  基于所提供代码，按要求实现改进的洗车仿真程序。具体要求如下。   * 1. Do not restrict the capacity of the car wash station.   不限制洗车服务台的容量。在这种情况下，洗车程序的终止条件可根据需要自行设定，例如可设定清洗车辆的数量上限，或者设定汽车到达时间的上限。   * 1. The inter-arrival time should be generated from an exponential distribution randomly. Let be the inter-arrival time, then follows an exponential distribution with cumulative distribution function   The parameter represents the arrival rate, which equals the reciprocal of the mean inter-arrival time, that is,  The arrival rate should be provided by the user.  汽车的到达时间间隔 服从指数分布，在仿真时需根据指数分布进行随机采样。为此，需要用户设定汽车的到达率，即平均到达时间间隔的倒数：  这样，到达时间间隔满足的指数分布具有累积分布函数：   * 1. The service time for each car should also be generated from an exponential distribution randomly via   The parameter represents the service rate, which equals the reciprocal of the mean service time, that is,  The service rate , or equivalently, the mean service time should be provided by the user. Note that mean service time should be less than mean inter-arrival time, which means should be greater than  汽车的服务时长也服从指数分布，在仿真时同样需根据指数分布进行采样。为此，需由用户设定服务台的服务率，即平均洗车时长的倒数：  这样，洗车时长服从的累积分布函数为  注意在你的设定中平均服务时长应该小于平均到达时间间隔，这意味着 >   * 1. To generate a random number satisfying exponential distributions, you could adopt the formula:   for inter-arrival time; or  for service time, with , a random number sampled from (0, 1) by uniform distribution.  为对到达时间间隔和洗车时长进行指数分布采样，可分别借助公式  和  进行计算，其中，, 是通过均匀分布产生的(0, 1)范围内的随机数。   * 1. Run multiple rounds of simulation, and each time record the obtained mean average waiting time and maximal queue length. See if your calculated average waiting time equals the theoretical value:   In this way you can certify the correctness of your simulation program.  进行多轮仿真实验，记录每次获取的所有汽车平均等待时长和最大等待队列长。计算分析你获取的平均等待时长是否与理论数值  接近。由此可验证你所编写仿真程序的正确性。  Task1：  实验思路：  本实验旨在改进一个洗车仿真程序，以解决现实世界中的排队问题。改进措施包括不限制洗车站的服务台容量、随机生成车辆到达时间间隔和服务时间，这些都基于指数分布，并且允许用户设定到达率和服务率。  详细实验步骤：   * 不限制服务台容量：移除原有洗车站的容量限制，改为根据实际需求设置终止条件，如最大模拟车辆数或最晚到达时间。 * 使用指数分布随机生成到达时间间隔：根据用户输入的平均到达率（即单位时间内预计到达的车辆数目），利用指数分布随机生成每辆车之间的到达时间间隔。 * 使用指数分布随机生成服务时间：同样地，根据用户提供的平均服务率（即单位时间内可以完成的服务次数），用指数分布来确定每辆车的服务时间。 * 双服务台系统：引入两个独立的服务台，当有新车到达时，优先选择空闲的服务台提供服务；如果两个服务台都在工作，则将车辆加入等待队列中，直到有服务台可用为止。 * 记录并计算统计指标：在仿真过程中，记录每个车辆的到达时间、开始服务时间和离开时间等信息，并据此计算平均等待时间、最大队列长度等性能指标。   核心代码如下(car.h, washCmp.h, car.cpp, washCmp.h, main.cpp)：  car.h:  #ifndef CAR\_H  #define CAR\_H  class Car  {  private:  double arrivalTime;  double departureTime;  double waitingTime;  double serviceTime;  public:  Car();  Car(double arrivalT);  double getArrivalTime();  double getDepartureTime();  double getWaitingTime();  double getServiceTime();  void setDepartAndWaitTime(double currTime, double serviceTime);  void printCarDeparture();  void printCarArrival();  };  #endif  car.cpp:  #include "car.h"  #include <iostream>  Car::Car()  {  arrivalTime = 0;  departureTime = 0;  waitingTime = 0;  }  Car::Car(double arrivalT)  {  arrivalTime = arrivalT;  departureTime = 0; //to be set later  waitingTime = 0; //to be set later  serviceTime = 0; //to be set later  }  double Car::getArrivalTime()  {  return arrivalTime;  }  double Car::getDepartureTime()  {  return departureTime;  }  double Car::getWaitingTime()  {  return waitingTime;  }  double Car::getServiceTime()  {  return serviceTime;  }  void Car::setDepartAndWaitTime(double startServiceTime, double serviceTime)  {  this->serviceTime = serviceTime;  departureTime = startServiceTime + serviceTime;  waitingTime = startServiceTime - arrivalTime; // Calculate waiting time  }  void Car::printCarDeparture()  {  std::cout << "A car is washed and departs!\n";  std::cout << "\tArrival time: " << arrivalTime << std::endl;  std::cout << "\tDeparture time: " << departureTime << std::endl;  std::cout << "\tWaiting time: " << waitingTime << std::endl;  std::cout << "\tService time: " << serviceTime << std::endl;  }  void Car::printCarArrival()  {  std::cout << "A car arrives!\n";  std::cout << "\tArrival time: " << arrivalTime << std::endl;  }  washCmp.h:  #ifndef WASH\_H  #define WASH\_H  #include <queue>  #include "car.h"  class WashCmp  {  private:  std::queue<Car> carQueue;  int numServedCars;  double totalWaitingTime;  int numCarsToSimulate;  double arrivalTimeNow; //previous arrival time  double lambda; // Arrival rate (lambda)  double mu; // Service rate (mu)  double getNextArrival(); // Generate next arrival time  double getServiceTime(); // Generate next service time  void processArrivalNonEmptyQ(double currTime); //a car arrives and the waiting queue is not empty  void processArrivalEmptyQ(double currTime); //a car arrives and the waiting queue is empty  void processDeparture();  void processRemain(); //no more arriving cars, process the remaining cars in the waiting queue  public:  WashCmp(double lambda, double mu, int numCars);  void simulation();  void printStatistic();  };  #endif  washCmp.cpp:  #include "washCmp.h"  #include <iostream>  #include <cstdlib>  #include <ctime>  #include <cmath>  WashCmp::WashCmp(double lambda, double mu, int numCars)  {  this->lambda = lambda;  this->mu = mu;  this->numCarsToSimulate = numCars;  totalWaitingTime = 0;  numServedCars = 0;  arrivalTimeNow = 0;  srand(time(0));  }  double WashCmp::getNextArrival()  {  double p = (double)rand() / RAND\_MAX;  double deltaT = (-1 / lambda) \* log(1 - p);  return arrivalTimeNow += deltaT;  }  double WashCmp::getServiceTime()  {  double q = (double)rand() / RAND\_MAX;  double serviceTime = (-1 / mu) \* log(1 - q);  return serviceTime;  }  void WashCmp::processArrivalEmptyQ(double arrivalTime)  {  Car arrivedCar = Car(arrivalTime);  arrivedCar.setDepartAndWaitTime(arrivalTime,getServiceTime()); //set the departure and waiting time of the arrived car  arrivedCar.printCarArrival(); //print the arrival information of the arrived car  carQueue.push(arrivedCar); //set the arrival time of the arrived car  }  void WashCmp::processArrivalNonEmptyQ(double arrivalTime)  {  Car arrivedCar = Car(arrivalTime);  arrivedCar.printCarArrival(); //print the arrival information of the arrived car  carQueue.push(arrivedCar); //set the arrival time of the arrived car  }  void WashCmp::processDeparture()  {  double currTime = carQueue.front().getDepartureTime(); //get the current time  totalWaitingTime += carQueue.front().getWaitingTime(); //update statistics  numServedCars++; //update statistics  carQueue.front().printCarDeparture(); //print departure information  carQueue.pop(); //departs  if (!carQueue.empty()) //set the departure and waiting time of  carQueue.front().setDepartAndWaitTime(currTime,getServiceTime()); //the current front car in the queue  }  void WashCmp::processRemain() //wash the remaining cars in the queue  {  while (!carQueue.empty())  processDeparture();  }  void WashCmp::simulation()  {  double arrivalTime = getNextArrival(); //get the first arrival time  while(numServedCars < numCarsToSimulate)  {  if (carQueue.empty()) //queue empty, process arrival  {  processArrivalEmptyQ(arrivalTime);  arrivalTime = getNextArrival(); //get next arrival  }  else if (arrivalTime < carQueue.front().getDepartureTime()) //arrival first, process arrival  {  processArrivalNonEmptyQ(arrivalTime);  arrivalTime = getNextArrival(); //get next arrival  }  else //departure first or of the same time, process departure  {  processDeparture(); //no need to get next arrival  }  }  //no more arrival, process the remaining cars in the queue  processRemain();  }  void WashCmp::printStatistic()  {  std::cout << "Number of served cars: " << numServedCars << std::endl;  std::cout << "Total waiting time: " << totalWaitingTime << std::endl;  if(numServedCars > 0)  std::cout << "The average waiting time is: " << (double)totalWaitingTime / numServedCars << std::endl;  else  std::cout << "The average waiting time is not applicable!" << std::endl;  // Calculate theoretical average waiting time  double thAvgWaitTime = lambda / (mu \* (mu - lambda));  std::cout << "The theoretical average waiting time is: " << thAvgWaitTime << std::endl;  }  main.cpp:  #include "car.h"  #include "washCmp.h"  #include <iostream>  using namespace std;  int main()  {  double lambda, mu;  int numCarsToSimulate;  // User inputs for arrival rate, service rate, and number of cars  cout << "Please input the mean arrival rate: ";  cin >> lambda;  cout << "Please input the mean service rate: ";  cin >> mu;  cout << "Please input the number of cars to be simulated: ";  cin >> numCarsToSimulate;  if (mu <= lambda)  {  cout << "The service rate must be greater than the arrival rate." << endl;  system("pause");  return 0;  }  // Create WashCmp object and start simulation  WashCmp cmp(lambda, mu, numCarsToSimulate);  cmp.simulation();  cmp.printStatistic();  system("pause");  return 0;  } | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）（此部分应包含实验结果，对实验结果的分析，实验收获的总结，实验中存在问题的讨论等；另外，需要回答一下如下思考题：   1）假设你仿真获得了一些洗车服务的指标，如平均等待时长，最大队列长度，等等，发现这些指标不满足洗车公司的设计需求，不能为顾客提供较好的服务。这时，一种改进的策略是增设一个服务台，用两个服务台同时提供汽车清洗服务。那么，怎么修改你的程序能够为这种双服务台系统进行仿真分析？可以简述你的设计思路，也可以编写代码验证你的思路。）  测试数据及结果：  数据1：    结果：    数据2：    结果：    数据3：    结果：    实验结果分析：  通过对三个不同的测试数据进行模拟，我们可以总结出以下几点：  模拟结果与理论计算值之间的差异相对较小，表明我们的模拟程序能够有效地反映排队系统的行为。在实验 #1 中，模拟得到的平均等待时间略低于理论值（误差为 0.46），而在实验 #2 中，模拟值几乎与理论值一致（误差仅为 0.02）。在实验 #3 中，模拟结果低于理论值（误差为 0.17），这可能是因为系统服务能力较强，排队时间较短。  总体而言，模拟中的误差较小，且趋势符合预期。这表明系统能够较好地处理不同到达率和服务率下的车辆，且模拟结果的随机性与理论预期相符。然而，模拟结果与理论值的差异主要受到到达率、服务率及系统容量等因素的影响，尤其是在服务率远大于到达率时，系统表现更为高效，平均等待时间较短。  实验总结：  通过模拟结果与理论值的对比，验证了模拟程序的正确性。大多数情况下，模拟的平均等待时间与理论值非常接近，说明模拟程序的设计是合理的。模拟中使用了指数分布来生成到达时间和服务时间，这种方式符合现实中车辆到达和服务的实际情况。指数分布在模拟中的使用帮助我们合理地处理随机到达和服务时间。本实验通过模拟车辆到达、排队和服务过程，帮助我们更好地理解了排队论中的一些基本概念，如平均等待时间、服务率、到达率等，并提供了一个简单的排队模型用于实际应用分析。实验中不仅验证了理论模型的准确性，还使我们意识到实际模拟中可能会受到多种因素的影响（例如随机性）。在模拟中，如果调整了不同的参数（例如服务率和到达率），结果会发生较大变化。  实验中存在问题的讨论：  在某些测试中，模拟的服务车辆数量超过了预期。这种现象可能是因为洗车站服务效率的不同变化导致的，可能需要在代码中加入检查机制，以确保模拟结果和实际情况相符。尽管模拟程序尽可能地接近理论模型，但由于引入了随机性，模拟结果与理论值之间的差距可能会随着实验的进行而略有变化。尽管这种差距通常较小，但可以通过增加模拟的轮数或样本量来提高结果的稳定性和准确性。在部分实验中出现了“等待时间为零”的情况，这在模拟中是合理的，但如果需要更精确地模拟排队行为，可能需要通过调整到达率和服务率的设置来增强排队过程的表现。实验结果的准确性可能受到模拟次数的影响。较少的模拟次数可能导致不准确的统计数据，因此建议增加模拟的车队数量或运行时间，进一步增加实验结果的稳定性。  思考题：  为了改进当前的洗车系统，以便模拟 双服务台 的情况，我们需要对现有的程序进行一些修改和扩展。   1. 设计思路：   在当前的单个服务台的基础上，增加第二个服务台。这样，车辆到达时可以选择两个服务台之一进行服务。服务台将以并行的方式提供服务，这意味着每个车的服务时间将由两个服务台的状态决定。需要设计一个队列来管理车辆的排队情况。当两个服务台都空闲时，车辆将立即开始接受服务。若两个服务台都在忙碌中，车辆将排队等待，直到有空闲的服务台。当车辆到达时，判断当前两个服务台的空闲情况。优先将车辆分配给空闲的服务台，如果两个服务台都忙碌，则将车辆放入队列中。需要记录两个服务台的服务状态、每个车的等待时间、离开时间等信息，并计算相关指标，如 平均等待时间、最大队列长度等。   1. 具体代码修改：   需要更新washCmp.h与washCmp.cpp。  我们需要为两个服务台分别维护独立的服务状态（空闲或忙碌）。每个服务台有一个对应的服务时间，车辆在到达后将选择空闲的服务台。   * 双服务台：我们新增了两个布尔变量 serviceStation1Free 和 serviceStation2Free 来表示两个服务台的空闲状态。在每次车辆到达时，程序会先检查两个服务台是否空闲，并优先为车辆分配空闲的服务台。如果两个服务台都忙碌，车辆会进入等待队列。 * 分配服务台：通过 processArrivalEmptyQ 和 processArrivalNonEmptyQ 两个方法来处理车辆的到达。根据当前服务台的空闲情况，分配车辆到空闲的服务台。 * 车队管理：如果两个服务台都忙碌，则车辆会进入等待队列，当有空闲服务台时，再从队列中取出车辆开始服务。 * 更新指标：在双服务台系统中，需要更新服务时间、等待时间等指标，计算总的等待时间、平均等待时间等，以便评估双服务台系统的性能。   通过这种修改，我们可以有效地模拟双服务台系统，并分析双服务台对洗车系统性能的影响。 | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |