

**Lab Report**

**实验报告**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2023-2024** |
| **Major**: | Software Engineering |
| **Class**: | 2022 |
| **Student Name**: | 朱昊 |
| **Student ID:** | 222022321062008 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Queue and Simulation  队列和仿真 | | | |
| Date | | Dec，2023 | 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. An exponential distribution has a distribution function   The parameter is based on the mean inter-arrrival time from the user’s input, that is,  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采样。为此，需设定汽车的到达率，即平均到达时间间隔的倒数：  这样，到达时间间隔满足的分布函数为   * 1. The service time for each car should be generated from an exponential distribution   ,  where  with mean service time provided by the user. Note that mean service time should be less than mean 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，洗车时长满足的分布函数为  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 sequence satisfying exponential distributions, you could adopt the formula:   for inter-arrival time; or  for service time, with *p*, *q* a random value in (0, 1) by uniform distribution.  为进行两个指数分布采样，可分别借助公式  和  其中，*p*和*q*是通过均匀分布产生的(0, 1)范围内的随机数。   * 1. Output the average waiting time and maximal queue length, using a large amount of simulation data. See if your calculated average waiting time equals:   多次进行仿真实验，每次均获取汽车的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数值  接近。  修改代码:  由于计算出来的车辆到达时间间隔是小数，所以要把在WashCmp里和car里定义的变量和函数里的形参都改成double      void WashCmp::simulation()//模拟  {  double arrivalTime = getNextArrival(); //get the next arrival time from keyboard input  //这里要把种子设置时间放到模拟函数里，这样才能确保随机数每次不相同  srand((unsigned int)time(0));  while (arrivalTime < 999.0)//设置当到达时间大于999则终止  {  if (carQueue.empty()) //queue empty, process arrival  {  processArrivalEmptyQ(arrivalTime);  arrivalTime = arrivalTime + getNextArrival(); //get next arrival  }  else if (arrivalTime < carQueue.front().getDepartureTime()) //arrival first, process arrival  { //如果到达时间小于队列第一个车的离开时间的这种情况  processArrivalNonEmptyQ(arrivalTime);  arrivalTime = arrivalTime + getNextArrival(); //get next arrival  }  else //departure first or of the same time, process departure  {//当到达时间大于第一个车的离开时间  processDeparture(); //no need to get next arrival  }  }  //no more arrival, process the remaining cars in the queue  processRemain();  }  double WashCmp::getNextArrival()//获取下一个汽车到达时间  {  double tempT;  double p= rand() / double(RAND\_MAX + 1);  double lambda = 0.3;  tempT = -(log(1 - p)) / lambda;  return tempT;  }  void WashCmp::processArrivalNonEmptyQ(double arrivalTime)//当队列不为空  {  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  //尝试更改最大队列长度的值  if (carQueue.size() > this->maxqueuelength)  {  maxqueuelength = carQueue.size();  }  }  void WashCmp::processDeparture()//  {  double currTime = carQueue.front().getDepartureTime();  totalWaitingTime += carQueue.front().getWaitingTime(); //update statistics  numServedCars++; //update statistics  carQueue.front().printCarDeparture(); //print departure information  carQueue.pop(); //departs  //如果队列还非空，就在计算新队列的第一个车的离开和等待时间  if (!carQueue.empty()) //set the departure and waiting time of  carQueue.front().setDepartAndWaitTime(currTime); //the current front car in the queue  }  void WashCmp::printCmpStatistic()  {  std::cout << "Number of served cars: " << numServedCars << std::endl;  std::cout << "Total waiting time: " << totalWaitingTime << std::endl;  std::cout << "The average waiting time is: " << (double)totalWaitingTime / numServedCars << std::endl;  std::cout << "最大等待队列长: " << this->maxqueuelength << std::endl;  } | | | | | |
| 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，需要回应一下如下思考题：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？）   实验结果在上面代码的基础下，设置arrivalTime不能大于999.0，且=0.5  。得出的程序运行结果如下图所示（指截取了部分）：      计算值=3 真实平均等待时间为3.43015  思考题：  在washcmp增加一个std::queue<Car> carQueue2;  修改代码为如下：  void WashCmp::simulation()//模拟  {  double arrivalTime = getNextArrival(); //get the next arrival time from keyboard input  srand((unsigned int)time(0));  while (arrivalTime < 10.0)  {  if (carQueue1.empty() || carQueue2.empty()) //queue empty, process arrival  {  processArrivalEmptyQ(arrivalTime);  arrivalTime = arrivalTime + getNextArrival(); //get next arrival  }  else if (arrivalTime < carQueue1.front().getDepartureTime() || arrivalTime < carQueue2.front().getDepartureTime()) //arrival first, process arrival  { //如果到达时间小于队列第一个车的离开时间的这种情况  processArrivalNonEmptyQ(arrivalTime);  arrivalTime = arrivalTime + getNextArrival(); //get next arrival  }  else //departure first or of the same time, process departure  {//当到达时间大于前两个车的离开时间  processDeparture(); //no need to get next arrival  }  }  //no more arrival, process the remaining cars in the queue  processRemain();  }  double WashCmp::getNextArrival()//获取下一个汽车到达时间  {  double tempT;  double p = rand() / double(RAND\_MAX + 1);  double lambda = 0.3;  tempT = -(log(1 - p)) / lambda;  /\*std::cout << "Please input the next arrival time (input 999 to terminate):\n";  std::cin >> tempT;\*/  return tempT;  }  void WashCmp::processArrivalEmptyQ(double arrivalTime)//当队列为空的时候的操作。  {  if (carQueue1.empty()) {  Car arrivedCar = Car(arrivalTime);  arrivedCar.setDepartAndWaitTime(arrivalTime); //set the departure and waiting time of the arrived car  cout << "use empty1:";  arrivedCar.printCarArrival(); //print the arrival information of the arrived car  carQueue1.push(arrivedCar); //set the arrival time of the arrived car  }  else {  Car arrivedCar = Car(arrivalTime);  arrivedCar.setDepartAndWaitTime(arrivalTime); //set the departure and waiting time of the arrived car  cout << "use empty2:";  arrivedCar.printCarArrival(); //print the arrival information of the arrived car  carQueue2.push(arrivedCar); //set the arrival time of the arrived car  }  }  void WashCmp::processArrivalNonEmptyQ(double arrivalTime)//当队列不为空  {  if (carQueue1.front().getDepartureTime() <= carQueue2.front().getDepartureTime()) {  if (arrivalTime > carQueue1.front().getDepartureTime()) {  //double currTime = carQueue1.front().getDepartureTime();  totalWaitingTime += carQueue1.front().getWaitingTime(); //update statistics  numServedCars++; //update statistics  carQueue1.front().printCarDeparture(); //print departure information  carQueue1.pop(); //departs  Car arrivedCar = Car(arrivalTime);  cout << "use noempty1:";  arrivedCar.printCarArrival();  carQueue1.push(arrivedCar);  carQueue1.front().setDepartAndWaitTime(arrivalTime);  }  else {  Car arrivedCar = Car(arrivalTime);  cout << "use noempty1:";  arrivedCar.printCarArrival(); //print the arrival information of the arrived car  carQueue1.push(arrivedCar); //set the arrival time of the arrived car  }  if (carQueue1.size() > this->maxqueuelength)  {  maxqueuelength = carQueue1.size();  }  }else{  if (arrivalTime > carQueue2.front().getDepartureTime()) {  totalWaitingTime += carQueue2.front().getWaitingTime(); //update statistics  numServedCars++; //update statistics  carQueue2.front().printCarDeparture(); //print departure information  carQueue2.pop(); //departs  Car arrivedCar = Car(arrivalTime);  cout << "use noempty2:";  arrivedCar.printCarArrival();  carQueue2.push(arrivedCar);  carQueue2.front().setDepartAndWaitTime(arrivalTime);  }  else {  Car arrivedCar = Car(arrivalTime);  cout << "use noempty2:";  arrivedCar.printCarArrival(); //print the arrival information of the arrived car  carQueue2.push(arrivedCar); //set the arrival time of the arrived car  }  if (carQueue2.size() > this->maxqueuelength)  {  maxqueuelength = carQueue2.size();  }  }  }  void WashCmp::processDeparture()//  {  if (carQueue1.front().getDepartureTime() <= carQueue2.front().getDepartureTime())  {  double currTime = carQueue1.front().getDepartureTime();  totalWaitingTime += carQueue1.front().getWaitingTime(); //update statistics  numServedCars++; //update statistics  carQueue1.front().printCarDeparture(); //print departure information  carQueue1.pop(); //departs  //如果队列还非空，就在计算新队列的第一个车的离开和等待时间  if (!carQueue1.empty()) //set the departure and waiting time of  carQueue1.front().setDepartAndWaitTime(currTime); //the current front car in the queue  }  else {  double currTime = carQueue2.front().getDepartureTime();  totalWaitingTime += carQueue2.front().getWaitingTime(); //update statistics  numServedCars++; //update statistics  carQueue2.front().printCarDeparture(); //print departure information  carQueue2.pop(); //departs  //如果队列还非空，就在计算新队列的第一个车的离开和等待时间  if (!carQueue2.empty()) //set the departure and waiting time of  carQueue2.front().setDepartAndWaitTime(currTime); //the current front car in the queue  }  }  void WashCmp::processRemain() //wash the remaining cars in the queue  {  while (!(carQueue1.empty() || carQueue2.empty())) {  if (carQueue1.empty()) {  double currTime = carQueue2.front().getDepartureTime();  totalWaitingTime += carQueue2.front().getWaitingTime(); //update statistics  numServedCars++; //update statistics  carQueue2.front().printCarDeparture(); //print departure information  carQueue2.pop(); //departs  //如果队列还非空，就在计算新队列的第一个车的离开和等待时间  if (!carQueue2.empty()) //set the departure and waiting time of  carQueue2.front().setDepartAndWaitTime(currTime); //the current front car in the queue  }  else {  double currTime = carQueue1.front().getDepartureTime();  totalWaitingTime += carQueue1.front().getWaitingTime(); //update statistics  numServedCars++; //update statistics  carQueue1.front().printCarDeparture(); //print departure information  carQueue1.pop(); //departs  //如果队列还非空，就在计算新队列的第一个车的离开和等待时间  if (!carQueue1.empty()) //set the departure and waiting time of  carQueue1.front().setDepartAndWaitTime(currTime); //the current front car in the queue  }  }  }  void WashCmp::printCmpStatistic()  {  std::cout << "Number of served cars: " << numServedCars << std::endl;  std::cout << "Total waiting time: " << totalWaitingTime << std::endl;  std::cout << "The average waiting time is: " << (double)totalWaitingTime / numServedCars << std::endl;  std::cout << "最大等待队列长: " << this->maxqueuelength << std::endl;  结果如下：也是且=0.5，    可以看到等待时间大大降低 | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |