

**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**: | 吴孜远 |
| **Student ID:** | 222022321062009 |
| **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。  **实验思路**：   1. 不限制排队数量，也就是说没有CAPACITY，也不会出现overflow。我们就以到达车辆的数量来作为程序终止条件，这就要在washCmp中加入数据成员totalCars，其值由用户输入，所以还需要一个配套公共函数getTotalCars()来设置totalCars。同时为了随时记录到达车辆的数量，我们还需要一个数据成员arrivedCars，在两个到达处理函数中都要自增。这时循环的判断条件变为(arrivedCars <= totalCars)，当到达车辆超出设定的数量时，跳出循环，处理完正在排队的车辆，终止程序。 2. 由于车辆到达时间间隔是在给出的指数函数中采样得出的，所以要修改getNextArrival()函数，由于采样的是时间间隔，根据“下一个到达时间=时间间隔+上一个车的到达时间”，我们还需要一个能在该函数中调用且储存上一个车到达时间的变量lastArrivalTime，声明为WashCmp的数据成员，它的值需要在两个到达函数中更新，也就是到达一辆车就要将其到达时间arrivalTime赋值给lastArrivalTime。 3. 服务时长也在给出的指数函数中采样得出，所以要更改setDepartAndWaitTime(int startServiceTime)函数，离开时间=开始时间+洗车时间。 4. 计算到达时间间隔（洗车时间）需要两个值，一个是平均到达率lambda（平均洗车率mu），一个是范围为（0，1）的随机数p（q）。随机数pq通过rand()/double(RAND\_MAX + 1)获取，在此之前要先在main函数初始化随机数种子。lambda和mu的值需要由用户输入，而lambda在WashCmp中使用，mu在Car中使用，这就要求这两个变量为extern全局变量，在头文件中声明，能够在不同源文件中访问（前提是引入了那个头文件并在当前文件中再次声明，这次声明不需要extern修饰符）。 5. 要获取最大列队数量，就要设置一个数据成员MaxWaitingCars，在每次到达车辆时都判断当前列队数量是否比当前MaxWaitingCars还要大。   **核心代码：**  car.cpp只改了一个函数，新建数据成员省略   1. void Car::setDepartAndWaitTime(double startServiceTime) 2. { 4. double p = rand() / double(RAND\_MAX + 1); 5. double t = -(log(1 - p)) / mu; 7. departureTime = startServiceTime + t; *//离开时间=开始时间+洗车时间* 8. waitingTime = startServiceTime - arrivalTime; *//等待时间=开始时间-到达时间* 9. }   Washcar整个文件几乎所有函数都有改动   1. #include "washCmp.h" 2. #include <iostream> 3. *//const int WashCmp::CAPACITY = 5 + 1;* 4. double lambda; 5. double mu; 6. WashCmp::WashCmp() 7. { 8. totalWaitingTime = 0; 9. numServedCars = 0; 10. totalCars = 0; 11. arrivedCars = 0; 12. MaxWaitingCars = 0; 13. lastArrivalTime = 0; 14. } 15. void WashCmp::simulation() 16. { 17. setLambda(); 18. setMu(); 19. int totalCars = getTotalCars(); 20. double arrivalTime = getNextArrival(); *//获取第一个到达时间* 21. while(arrivedCars <= totalCars) 22. { 23. if (carQueue.empty()) *//queue empty, process arrival* 24. { 25. processArrivalEmptyQ(arrivalTime); 26. arrivalTime = getNextArrival(); *//get next arrival* 27. } 28. else if (arrivalTime < carQueue.front().getDepartureTime()) *//arrival first, process arrival* 29. { 30. processArrivalNonEmptyQ(arrivalTime); 31. arrivalTime = getNextArrival(); *//get next arrival* 32. } 33. else *//departure first or of the same time, process departure* 34. { 35. *//若下一辆车到达时间远大于服务车离开时间，则每次都处理离开而不获取下一个到达时间，直到上一次的到达时间小于离开时间* 36. processDeparture(); *//no need to get next arrival* 37. } 38. } 39. *//no more arrival, process the remaining cars in the queue* 40. processRemain(); 41. } 42. void WashCmp::setLambda() 43. { 45. double meanInterArrivalTime; *//3* 46. std::cout << "input the mean inter arrival time:"; 47. std::cin >> meanInterArrivalTime; 48. lambda = 1 / meanInterArrivalTime; 49. *//std::cout << lambda << std::endl;* 50. } 51. void WashCmp::setMu() 52. { 53. double meanServiceTime; *//2* 54. std::cout << "input the mean service time:"; 55. std::cin >> meanServiceTime; 56. mu = 1 / meanServiceTime; 57. *//std::cout << mu << std::endl;* 58. } 59. double WashCmp::getNextArrival() 60. { 61. double p = rand() / double(RAND\_MAX + 1); 62. double t = - (log(1 - p)) / lambda; 63. return t + lastArrivalTime; *//下一个到达时间=时间间隔+上一个车的到达时间* 64. } 65. int WashCmp::getTotalCars() 66. { 67. int tempT; 68. std::cout << "Please input the total car number: "; 69. std::cin >> tempT; 70. std::cout << "\n"; 71. return tempT; 72. } 73. void WashCmp::processArrivalEmptyQ(double arrivalTime) 74. { 75. Car arrivedCar = Car(arrivalTime); 76. arrivedCar.setDepartAndWaitTime(arrivalTime); *//set the departure and waiting time of the arrived car* 77. arrivedCar.printCarArrival(); *//print the arrival information of the arrived car* 78. carQueue.push(arrivedCar); *//set the arrival time of the arrived car* 79. if (MaxWaitingCars < carQueue.size()) { 80. MaxWaitingCars = carQueue.size(); 81. } 82. arrivedCars++; *//计数到达车辆* 83. lastArrivalTime = arrivalTime; *//记录这一辆车的到达时间用于计算下一辆车的到达时间* 84. } 85. void WashCmp::processArrivalNonEmptyQ(double arrivalTime) 86. { 87. *// if (carQueue.size() < CAPACITY) //若还有空位，则加入等待* 88. *//{* 89. Car arrivedCar = Car(arrivalTime); 90. arrivedCar.printCarArrival(); *//print the arrival information of the arrived car* 91. carQueue.push(arrivedCar); *//set the arrival time of the arrived car* 92. *//每次加入都更新最大队列数* 93. if (MaxWaitingCars < carQueue.size()) { 94. MaxWaitingCars = carQueue.size(); 95. } 96. arrivedCars++; 97. lastArrivalTime = arrivalTime; 98. *//}* 99. *//else //the arrived car leaves* 100. *//std::cout << "OVERFLOW!!!\n";* 101. } 102. void WashCmp::processDeparture() 103. { 104. double currTime = carQueue.front().getDepartureTime(); *//get the current time* 105. totalWaitingTime += carQueue.front().getWaitingTime(); *//update statistics* 106. numServedCars++; *//update statistics* 107. carQueue.front().printCarDeparture(); *//print departure information* 108. carQueue.pop(); *//departs* 109. if (!carQueue.empty())*//set the departure and waiting time of* 110. *//下一辆车开始服务，currTime是上一辆车离开时间也是当前车的开始服务时间* 111. carQueue.front().setDepartAndWaitTime(currTime);   *//设置当前服务车的离开时间（未来）和等待时间* 112. } 113. void WashCmp::processRemain() *//wash the remaining cars in the queue* 114. { 115. while (!carQueue.empty()) 116. processDeparture(); 117. } 118. void WashCmp::printCmpStatistic() 119. { 120. *//std::cout << "Number of served cars: " << numServedCars << std::endl;* 121. std::cout << "Total waiting time: " << totalWaitingTime << std::endl; 122. std::cout << "The average waiting time is: " << (double)totalWaitingTime / numServedCars << std::endl; 123. std::cout << "Total Max Waiting Cars: " << MaxWaitingCars - 1 << std::endl; 124. std::cout << "Theoretical value of average waiting time：" << lambda / (mu \* (mu - lambda)); 126. } | | | | | |
| 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：**  测试5组数据，参数相同，此时平均等待时间的理论值为4  实验值平均值为3.778，大体接近            **思考题：**  **有两个服务台，且洗车时间每次是不同的，那就不能将队首作为服务台了，我们可以开辟一个服务台链表，将正在洗的车辆弹出放在里面，表示正在洗车。到达事件：先检查这个链表长度是否小于2；若是，则直接进入服务台；若否，则进入队列。若数组中两辆车的最小离开时间不小于到达时间，则处理离开事件：使离开时间最小的车离开，下一辆车进入。**  **核心代码如下：**   1. void WashCmp::simulation() 2. { 3. setLambda(); 4. setMu(); 5. int totalCars = getTotalCars(); 6. double arrivalTime = getNextArrival(); *//获取第一个到达时间* 8. while(arrivedCars <= totalCars) 9. { 10. if (carQueue.empty() && (washingBay.size() < 2)) *//queue empty, process arrival* 11. { 12. processArrivalEmptyQ(arrivalTime); 13. arrivalTime = getNextArrival(); *//get next arrival* 14. } 15. else if (arrivalTime < getMinDepartureTime() && (washingBay.size() == 2)) *//arrival first, process arrival* 16. { 17. processArrivalNonEmptyQ(arrivalTime); 18. arrivalTime = getNextArrival(); *//get next arrival* 19. } 20. else *//departure first or of the same time, process departure* 21. { 22. *//若下一辆车到达时间远大于服务车离开时间，则每次都处理离开而不获取下一个到达时间，直到上一次的到达时间小于离开时间* 23. processDeparture(); *//no need to get next arrival* 24. } 25. } 26. *//no more arrival, process the remaining cars in the queue* 27. processRemain(); 28. } 29. double WashCmp::getMinDepartureTime() 30. { 31. std::list<Car>::iterator itr = washingBay.begin(); 32. double a = (\*itr).getDepartureTime(); 33. double b = (\*++itr).getDepartureTime(); 34. return std::min(a, b); 35. } 36. void WashCmp::processArrivalEmptyQ(double arrivalTime) 37. { 38. Car arrivedCar = Car(arrivalTime); 39. arrivedCar.setDepartAndWaitTime(arrivalTime); *//set the departure and waiting time of the arrived car* 40. *//arrivedCar.printCarArrival(); //print the arrival information of the arrived car* 42. washingBay.push\_back(arrivedCar); 43. if (MaxWaitingCars < carQueue.size()) { 44. MaxWaitingCars = carQueue.size(); 45. } 46. arrivedCars++; *//计数到达车辆* 47. lastArrivalTime = arrivalTime; *//记录这一辆车的到达时间用于计算下一辆车的到达时间* 48. } 49. void WashCmp::processArrivalNonEmptyQ(double arrivalTime) 50. { 51. Car arrivedCar = Car(arrivalTime); 52. *//arrivedCar.printCarArrival(); //print the arrival information of the arrived car* 53. carQueue.push(arrivedCar); *//set the arrival time of the arrived car* 54. *//每次加入都更新最大队列数* 55. if (MaxWaitingCars < carQueue.size()) { 56. MaxWaitingCars = carQueue.size(); 57. } 58. arrivedCars++; 59. lastArrivalTime = arrivalTime; 60. } 61. void WashCmp::processDeparture() *//将最小的移除* 62. { 63. double currTime = getMinDepartureTime(); 64. std::list<Car>::iterator it = washingBay.begin(); 65. int i = 2; 66. while (i > 0) { 67. if ((\*it).getDepartureTime() == currTime) { 68. break; 69. } 70. it++; 71. i--; 72. } 74. totalWaitingTime += (\*it).getWaitingTime(); *//update statistics* 75. *//(\*it).printCarDeparture(); //print departure information* 76. washingBay.erase(it); 77. numServedCars++; *//update statistics* 78. if (!carQueue.empty()) {*//set the departure and waiting time of* 79. *//下一辆车开始服务，currTime是上一辆车离开时间也是当前车的开始服务时间* 80. carQueue.front().setDepartAndWaitTime(currTime);   *//设置当前服务车的离开时间（未来）和等待时间* 81. washingBay.push\_back(carQueue.front()); 82. carQueue.pop(); 83. } 84. }   **结果：**    **明显平均时间降低了非常多。** | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |