

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2020-2021** |
| **Major**: | Software Engineering |
| **Class**: | 2019 |
| **Student Name**: | 冯春霖 |
| **Student ID:** | 222019321062074 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Queue and Simulation | | | |
| Date | | Dec，2020 | Type | | □Confirmatory  √ Design  □Comprehensive |
| 1. **Objective & Requirements**    1. Understand the concept of container adapter    2. Know the implementation the queue container adapter in the STL    3. Grasp the use of queue container in a real application    4. Know the concept of simulation and can use simulation to solve a real problem    5. Know about the queueing theory and the exponential distribution theory | | | | | |
| 1. **Experimental environment (**platform and software**)**   Windows 7 (or higher versions) + Visual Studio 2010 (or higher versions) | | | | | |
| 1. **Experimental content and design** (Main Content, Procedure, Codes and Results)   Task 1  Improve the car wash simulation problem in the following ways based on the codes and slides sent to you:   * 1. Do not restrict the capacity of the car wash station.   2. 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,   * 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 larger than   * 1. To generate a sequence satisfying exponential distributions, you could adopt the formula:   or  with *p* a random value in (0, 1) by uniform distribution.   * 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:   Improved program is as follows:  In main.cpp:  int main()  {  srand((unsigned int)time(NULL));  WashCmp cmp;  cmp.simulation();  cmp.printCmpStatistic();  return 0;  }  Added srand() as random number seeds.  In car class:  class Car  {  private:  int arrivalTime;  int departureTime;  int waitingTime;  //const static int SERVICE\_T;  double serviceTime;  public:  Car();  Car(int arrivalT, double mu);  int getArrivalTime();  int getDepartureTime();  int getWaitingTime();  void setDepartAndWaitTime(int currTime);  void printCarDeparture();  void printCarArrival();  };  Pass mu as a parameter to the constructor with parameters, removes the constant SERVICE\_T  The implementation of the constructor with parameters:  Car::Car(int arrivalT, double mu)  {  arrivalTime = arrivalT;  departureTime = 0;  waitingTime = 0;  serviceTime = -1.0 / mu \* log(1 - rand() / (double)(RAND\_MAX + 1));  }  A random serviceTime is generated in constructor.  In WashCmp class:  class WashCmp  {  private:  std::queue<Car> carQueue;  const static int CAPACITY;  int numServedCars;  int totalWaitingTime;  double lambda, mu;  double arrivalTime;  int totalServiceNumber;  int maxQueueLength;  int getNextArrival(); //accept user input to get the next arrival time  void processArrivalNonEmptyQ(int currTime); //a car arrives and the waiting queue is not empty  void processArrivalEmptyQ(int 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();  void simulation();  void printCmpStatistic();  };  Added lambda, mu, totalServiceNumber and maxQueueLength as variables, change arrivalTime as member variables.  WashCmp::WashCmp()  {  totalWaitingTime = 0;  numServedCars = 0;  double temp;  std::cout << "Please input the mean arrival rate:\n";  std::cin >> temp;  lambda = 1.0 / temp;  std::cout << "Please input the mean service rate:\n";  std::cin >> temp;  mu = 1.0 / temp;  std::cout << "Please input the number of cars to be simulate:\n";  std::cin >> totalServiceNumber;  }  Some input and output statements are added to the constructor of WashCmp to initialize the variables.  void WashCmp::simulation()  {  arrivalTime = getNextArrival();  while (numServedCars != totalServiceNumber)  {  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();  }  Change the loop condition of the simulation function.  int WashCmp::getNextArrival()  {  int tempT;  //std::cout << "Please input the next arrival time (input 999 to terminate):\n";  tempT = arrivalTime - 1.0 / lambda \* log(1 - rand() / (double)RAND\_MAX);  return tempT;  }  Change the getNextArrival function to use a randomly generated variable as the return value.  void WashCmp::processArrivalEmptyQ(int arrivalTime)  {  Car arrivedCar = Car(arrivalTime, mu);  arrivedCar.setDepartAndWaitTime(arrivalTime); //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  if (carQueue.size() > maxQueueLength)  maxQueueLength = carQueue.size();  }  void WashCmp::processArrivalNonEmptyQ(int arrivalTime)  {  Car arrivedCar = Car(arrivalTime, mu);  arrivedCar.printCarArrival(); //print the arrival information of the arrived car  carQueue.push(arrivedCar); //set the arrival time of the arrived car  if (carQueue.size() > maxQueueLength)  maxQueueLength = carQueue.size();  }  Change the processArrival function to remove its queue length limit and record the maximum queue size.  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 << "Max length of queue is: " << maxQueueLength << std::endl;  std::cout << "lambda / (mu \* (mu - lambda)): " << lambda / (mu \* (mu - lambda)) << std::endl;  }  Change the printCmpStatistics function to output the values of all relevant variables.  Test in main.    It can be seen that the results of the function simulation are in line with expectations. | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   Through this experiment, I have gained some understanding of container adapters, and at the same time, mastered some knowledge about queues and learned practical program simulation to solve some practical problems. In the process I also learned to use random functions to generate random numbers that obey exponential distribution. There are no remaining problems in this experiment. | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |