Joshua Wright

10/03/2015

COP 4534

A Queue Simulation

|  |  |  |
| --- | --- | --- |
| File | Functions | Purpose |
| Calc.h | None | Function decelerations, struct decelerations. User interface for calc.c |
| Calc.c | 1. Perc\_Idle\_Time(float avg\_Arrivals, float avg\_Num\_Serv, int serv\_Chan) 2. Avg\_Num\_n\_System(float avg\_Arrivals, float avg\_Num\_Serv, int serv\_Chan, float perc\_Idle\_Time) 3. Avg\_Time\_n\_System(float avg\_Arrivals, float avg\_Num\_n\_System) 4. Avg\_Cust\_n\_Queue(float avg\_Num\_n\_System, float avg\_Arrivals, Float avg\_Num\_Serv) 5. Avg\_Time\_Wait\_n\_Queue(float avg\_Cust\_n\_Queue, float avg\_Arrivals) 6. Util\_Fac(float avg\_Arrivals, float avg\_Num\_Serv, int serv\_Chan) 7. Factorial(int i) | Struct definition:   1. Finds the Percent Idle Time 2. Finds the Avg Number in the system 3. Finds the average time in the System 4. Finds the Avg Customer in the Queue 5. Finds the Avg Time waiting in the queue 6. Finds the util factor 7. Helper function to find the factorial   ALL EQUATIONS ARE FOR CALCULATED RESULTS ONLY. |
| Customer.h | None | Function decelerations, struct decelerations. User interface for customer.c |
|  |  |  |
| Customer.c | 1. freeCustomer(customerT \*customer) 2. cust\_Switch(customer \*\*tempA, customer \*\* tempB) 3. cust\_Dup(customerT \*from, customerT \*going) | Struct Definition:   1. allocated memory for customer and sets times to -1.0. Sets nextCust to NULL 2. switches the index of 2 customers positions 3. duplicates the times of one customer to another |
|  |  |  |
| pQueue.h | None | Function decelerations, struct decelerations. User interface for pQueue.c |
|  |  |  |
| pQueue.c | 1. createPQ ( ) 2. enqueuePQ(customerT \*customer, pQueueT \*pQueue) 3. dequeuePQ(pQueueT \*pQueue) 4. upPQ(pQueueT \*pQueue, int length) 5. downPQ(pQueueT \*pQueue, int length) 6. pQfull(pQueueT \*pQueue) 7. pQempty(pQueueT \*pQueue) 8. freePQ(pQueueT \*pQueue) | Struct Definition:   1. malloc memory for newQueue, set the length to 0, and set newQueue->customers[i] to NULL 2. puts a customer into the pQueue and perculates up 3. takes a customer from the pQueue, copies the information, down perculates 1, and decrements pQueue->length 4. perculates the customer up depending on the PQtime 5. perculates the customer down starting at the half 6. checks to see if the PQ is full 7. checks to see if the PQ is empty 8. frees the PQ |
|  |  |  |
| Queue.h | None | Function decelerations, struct decelerations. User interface for queue.c |
|  |  |  |
| Queue.c | 1. createQueue( ) 2. enqueuer(customerT \*customer, queueT \*queue) 3. dequeuer(queueT \*queue) 4. qEmpty(queueT \*queue) 5. freeQueue(queueT \*queue) | Struct Definition:   1. mallocs memory for queue, sets length to 0, and sets head of queue to NULL 2. adds a customer to the fifo queue and increments queue length 3. removes customer from fifo queue and decrements the length 4. checks to see if the fifo queue is empty 5. free’s the allocated memory for the fifo queue |
|  |  |  |
| qSim.c | 1. genRandInterval(int avg) 2. getData(calcT \*calc) 3. setData(calcT \*calc) 4. calcData(calcT \*calc, equationsT \*equations) 5. primePump(pQueueT \*pQueue, calcT \*calc) 6. simResults(calcT \*calc, equationsT \*equations) 7. doWork(pQueueT \*pQueue, queueT \*queue, calcT \*calc, equationsT \*equations) 8. equationsInit(equationsT \*equations) 9. printStats(equationsT \*equations) | 1. generates a random number between 0 and 1 2. retrieves the data from the file runSimulation.txt 3. sets certain variables to their opposites for the beginning. 4. Runs each of the equations to get the calculated results for the problems 5. Fill the PQ with customers to start the event 6. Calculate the even drive results to test with the calculated results. 7. The even function, doing certain things if the customer is arriving or if the customer is departing 8. Initializing certain equation variables to 0 9. Prints out the calculated results alongside the even driven results. |

Data Structures:

typedef struct calc{

double avg\_Arrivals;

double avg\_Num\_Serv;

int avail\_Serv;

double last\_Arrival;

double last\_Dep;

double startIdle;

int serv\_Chan;

int num\_Of\_Arrivals;

int remaining\_Cust;

}calcT;

typedef struct Customer{

double PQtime;

double arrivalTimes;

double startOfServiceTime;

double departureTime;

struct Customer \*nextCust;

}customerT;

typedef struct pQueue{

int length;

customerT \*customers[max\_length];

}pQueueT;

typedef struct queue{

int length;

customerT \*head;

} queueT;

typedef struct Equations{

double perc\_Idle\_Time\_Equation, perc\_Idle\_Time\_Results;

double avg\_Num\_n\_System\_Equation;

double avg\_Time\_n\_System\_Equation, avg\_Time\_n\_System\_Results;

double avg\_Cust\_n\_Queue\_Equation;

double avg\_Time\_Wait\_n\_Queue\_Equation, avg\_Time\_Wait\_n\_Queue\_Results;

double util\_Fac\_Equation, util\_Fac\_Results;

double queue\_Total\_Time;

double system\_Total\_Time;

double idle\_Total\_Time;

double prob\_Of\_Waiting\_For\_Server;

int waited\_Count;

}equationsT;