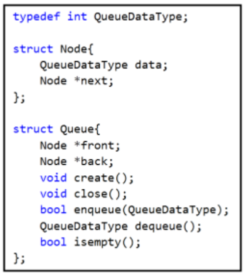
Data Structures 2012 Fall

Practice Session 5

Contents

- Queue Simulation with one server.

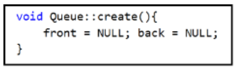
Queue Structure



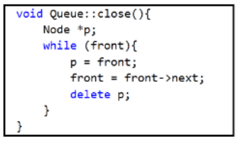
```
typedef int QueueDataType;

struct Node{
    QueueDataType data;
    Node *next;
};

struct Queue{
    Node *front;
    Node *back;
    void create();
    void close();
    bool enqueue(QueueDataType);
    QueueDataType dequeue();
    bool isempty();
};
```

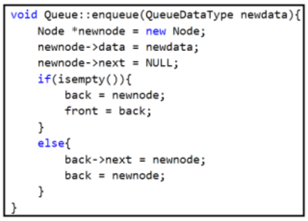


```
void Queue::create(){
    front = NULL; back = NULL;
}
```

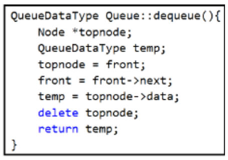


```
void Queue::close(){
    Node *p;
    while (front){
        p = front;
        front = front->next;
        delete p;
    }
}
```

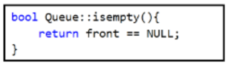
Queue Structure



```
void Queue::enqueue(QueueDataType newdata){
    Node *newnode = new Node;
    newnode->data = newdata;
    newnode->next = NULL;
    if(isempty()){
        back = newnode;
        front = back;
    }
    else{
        back->next = newnode;
        back = newnode;
    }
}
```

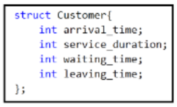


```
QueueDataType Queue::dequeue(){
    Node *topnode;
    QueueDataType temp;
    topnode = front;
    front = front->next;
    temp = topnode->data;
    delete topnode;
    return temp;
}
```

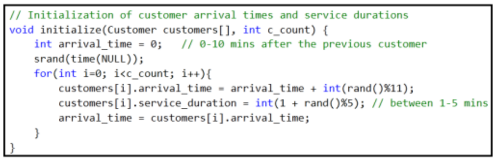


```
bool Queue::isempty(){
    return front == NULL;
}
```

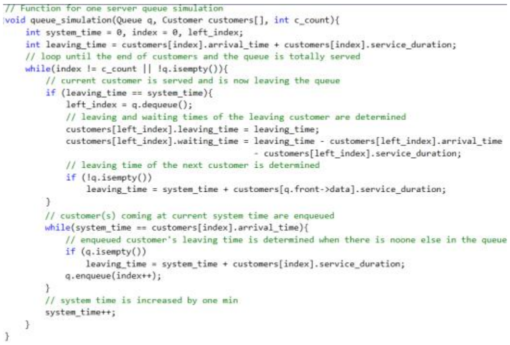
Customer Structure and Initialization of Arrival Time & Service Duration



```
struct Customer{
    int arrival_time;
    int service_duration;
    int waiting_time;
    int leaving_time;
};
```



```
// Initialization of customer arrival times and service durations
void initialize(Customer customers[], int c_count) {
    int arrival_time = 0;    // 0-10 mins after the previous customer
    srand(time(NULL));
    for(int i=0; i<c_count; i++){
        customers[i].arrival_time = arrival_time + int(rand()%11);
        customers[i].service_duration = int(1 + rand()%5); // between 1-5 mins
        arrival_time = customers[i].arrival_time;
    }
}
```



```
// Function for one server queue simulation
void queue_simulation(Queue q, Customer customers[], int c_count){
    int system_time = 0, index = 0, left_index;
    int leaving_time = customers[index].arrival_time + customers[index].service_duration;
    // loop until the end of customers and the queue is totally served
    while(index != c_count || !q.isempty()){
        // current customer is served and is now leaving the queue
        if (leaving_time == system_time){
            left_index = q.dequeue();
            // leaving and waiting times of the leaving customer are determined
            customers[left_index].leaving_time = leaving_time;
            customers[left_index].waiting_time = leaving_time - customers[left_index].arrival_time
                                                  - customers[left_index].service_duration;
            // leaving time of the next customer is determined
            if (!q.isempty())
                leaving_time = system_time + customers[q.front->data].service_duration;
        }
        // customer(s) coming at current system time are enqueued
        while(system_time == customers[index].arrival_time){
            // enqueued customer's leaving time is determined when there is noone else in the queue
            if (q.isempty())
                leaving_time = system_time + customers[index].service_duration;
            q.enqueue(index++);
        }
        // system time is increased by one min
        system_time++;
    }
}
```

Printing out Customer Time Information on the Screen

```
// Function to print out time and duration information for customers
void print_results(Customer customers[], int c_count){
    cout << "Customer  num:\tArrival time:\tWaiting time:\tService dur.:\tLeaving time:\n";
    cout << "--------------\t-------------\t-------------\t-------------\t-------------\n";
    for(int i=0; i<c_count; i++)
        cout << i+1 << "\t\t"
             << customers[i].arrival_time << "\t\t"
             << customers[i].waiting_time << "\t\t"
             << customers[i].service_duration << "\t\t"
             << customers[i].leaving_time << endl;
}
```

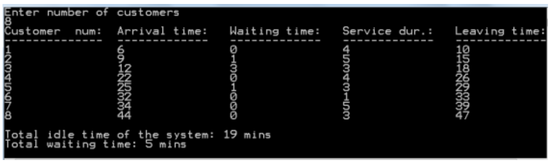
Determining Total Waiting & Idle Times

```
// Function to determine total idle and waiting times
void system_performance(Customer customers[], int c_count){
    int total_idle_time = 0;
    int total_waiting_time = 0;
    int leaving_time = 0;
    int i = 0;
    while(i < c_count){
        if(customers[i].arrival_time > leaving_time)
            total_idle_time = total_idle_time + customers[i].arrival_time - leaving_time;
        total_waiting_time = total_waiting_time + customers[i].waiting_time;
        leaving_time = customers[i].leaving_time;
        i++;
    }
    cout << "\nTotal idle time of the system: " << total_idle_time << " mins\n";
    cout << "Total waiting time: " << total_waiting_time << " mins\n";
}
```

Test Program

```
int main(){
    // Creation and initialization of customers
    int c_count;
    cout << "Enter number of customers " << endl;
    cin >> c_count;
    struct Customer* customers = new Customer[c_count];
    initialize(customers, c_count);
    // Creation of customer queue
    Queue q;
    q.create();
    // Queue simulation and printing out the results
    queue_simulation(q, customers, c_count);
    print_results(customers, c_count);
    system_performance(customers, c_count);
    // Allocated memory for customers is given back
    delete [] customers;
    return EXIT_SUCCESS;
}
```

Screenshot



```
Enter number of customers
8
Customer  num:  Arrival time:   Waiting time:   Service dur.:   Leaving time:
--------------  -------------   -------------   -------------   -------------
1               6               0               4               10
2               9               1               5               15
3               12              3               3               18
4               22              0               4               26
5               25              1               3               29
6               32              0               1               33
7               34              0               5               39
8               44              0               3               47

Total idle time of the system: 19 mins
Total waiting time: 5 mins
```