CSCI 301 Computer Science II Summer 2023

Assignment 7 **Simulation of a Bank System**

Due Date: 11:50 pm on July 26, Wednesday

# Introduction

This program consists of gathering the arrivals and process of a bank customer and then displaying the departure time. The program counts the customers and keeps track of their cumulative waiting time. The user input a file that contains arrivals and transactions time, then the program will process the departure and finally display the total number of customers processed and the Average amount of time spent waiting. We have used a queue and priority queue to successfully implement this project.

# Data Structure

In this project, we have used the following data structure in order to successfully design and implement the program. A priority queue to store the events, booleans for the operators, and a few integers.

# Functions

We have design a few functions to successfully implement the program among them we have:

* operator >
* operator<
* operator==
* Operator!
* bankSimulation

# Structure chart of the main program



# Code list

/\*\*

\* problem description: Implement an event-driven simulation of a bank maintaining the arrival

\* events and the departure events.

\* Name: Algassimou Diallo

\* startID: el8524jv

\* Instructor: Jie Meichsner

\* Due date: 07/26/2023

\*/

#include <iostream>

#include <fstream>

#include <queue>

#ifndef EVENT\_H

#define EVENT\_H

using namespace std;

/\*\*

\* @brief Class to simulate an arrival or departure event.

\*/

class Event {

public:

char type; // 'A' for arrival event, 'D' for departure event

int time; // The time of the event (arrival or departure)

int transTime; // Transaction time for the customer

/\*\*

\* @brief Constructor for Event class.

\* @param eventType The type of the event (arrival or departure)

\* @param eventTime The time of the event (arrival or departure)

\* @param transactionTime The transaction time for the customer

\*/

Event(char eventType, int eventTime, int transactionTime) : type(eventType), time(eventTime), transTime(transactionTime) {}

// Define comparison operators for priority queue (smallest time has highest priority)

bool operator>(const Event& other) const {

return time > other.time;

}

bool operator<(const Event& other) const {

return time < other.time;

}

bool operator==(const Event& other) const {

return time == other.time;

}

bool operator!=(const Event& other) const {

return time != other.time;

}

};

/\*\*

\* @brief Function to simulate the bank's event-driven simulation.

\* @param inputFile The name of the input file containing arrival and transaction times.

\*/

void bankSimulation(const string& inputFile) {

priority\_queue<Event, vector<Event>, greater<Event> > eventQueue; // Priority queue to store events

int totalTime = 0; // Cumulative waiting time

int totalCustomers = 0; // Total number of customers

ifstream inFile(inputFile);

if (!inFile.is\_open()) {

cerr << "Error opening file: " << inputFile << endl;

return;

}

// Read the input file and add arrival events to the event queue

int arrivalTime, transactionTime;

while (inFile >> arrivalTime >> transactionTime) {

eventQueue.push(Event('A', arrivalTime, transactionTime));

}

inFile.close();

// Process the events in the event queue

while (!eventQueue.empty()) {

Event currentEvent = eventQueue.top();

eventQueue.pop();

if (currentEvent.type == 'A') {

// Arrival event

cout << "Processing an arrival event at time: " << currentEvent.time << endl;

// Add the customer's transaction time to the total waiting time

totalTime += currentEvent.transTime;

// Increment the total number of customers

totalCustomers++;

// Create the corresponding departure event and add it to the event queue

eventQueue.push(Event('D', currentEvent.time + currentEvent.transTime, 0));

} else {

// Departure event

cout << "Processing a departure event at time: " << currentEvent.time << endl;

}

}

// Compute the average waiting time

double averageWaitingTime = static\_cast<double>(totalTime) / totalCustomers;

// Display the final statistics

cout << "Simulation ends." << endl;

cout << "Final Statistics:" << endl;

cout << "Total number of customers processed: " << totalCustomers << endl;

cout << "Average amount of time spent waiting: " << averageWaitingTime << endl;

}

#endif

—-----------------------------------------------------------------—----------------------------------------------

/\*\*

\* problem description: Implement an event-driven similation of a bank maintaining the arrival

\* events and the departure events.

\* Name: Algassimou Diallo

\* startID: el8524jv

\* Instructor: Jie Meichsner

\* Due date: 07/26/2023

\*/

#include <iostream>

#include <fstream>

#include <queue>

#include "event.h"

int main() {

string fileName;

cout << "Enter the name of the file: ";

cin >> fileName;

// Call the bankSimulation function with the input file name

bankSimulation(fileName);

return 0;

}

# User Document

To successfully run this program, the program will request the user to enter the file to be processed. The user will enter the file name (e.g. file1.txt). This file contains the arrival time in the first column and the transaction time in the second column. Then, the program will precess the arrival and departure time and then display the total number of customers and also the average amount of time spent waiting.

The files of this project are located in the directory el8524jv/csci301/project7. To successfully compile this program in centOS, the user must run the following script:

g++ -std=c++11 -g main.cpp -o a.out. Then you run ./a.out.

Test cases

| Tests | Inputs | outputs |
| --- | --- | --- |
| 1. Ascendent ordered arrivals time (file1.txt) | Arrival transaction  1 5  2 5  4 5  20 5  22 5  24 5  26 5  28 5  30 5  88 3 | Processing an arrival event at time: 1  Processing an arrival event at time: 2  Processing an arrival event at time: 4  Processing a departure event at time: 6  Processing a departure event at time: 7  Processing a departure event at time: 9  Processing an arrival event at time: 20  Processing an arrival event at time: 22  Processing an arrival event at time: 24  Processing a departure event at time: 25  Processing an arrival event at time: 26  Processing a departure event at time: 27  Processing an arrival event at time: 28  Processing a departure event at time: 29  Processing an arrival event at time: 30  Processing a departure event at time: 31  Processing a departure event at time: 33  Processing a departure event at time: 35  Processing an arrival event at time: 88  Processing a departure event at time: 91  Simulation ends.  Final Statistics:  Total number of customers processed: 10  Average amount of time spent waiting: 4.8 |
| 1. Unordered arrival time(file2.txt) | Arrival transaction  1 3  4 5  29 2  19 4  30 7 | Processing an arrival event at time: 1  Processing an arrival event at time: 4  Processing a departure event at time: 4  Processing a departure event at time: 9  Processing an arrival event at time: 19  Processing a departure event at time: 23  Processing an arrival event at time: 29  Processing an arrival event at time: 30  Processing a departure event at time: 31  Processing a departure event at time: 37  Simulation ends.  Final Statistics:  Total number of customers processed: 5  Average amount of time spent waiting: 4.2 |
| 1. All transactions time are 0 (file3.txt) | Arrival transaction  1 0  4 0  19 0  29 0 | Processing an arrival event at time: 1  Processing a departure event at time: 1  Processing an arrival event at time: 4  Processing a departure event at time: 4  Processing an arrival event at time: 19  Processing a departure event at time: 19  Processing an arrival event at time: 29  Processing a departure event at time: 29  Simulation ends.  Final Statistics:  Total number of customers processed: 4  Average amount of time spent waiting: 0 |
| 1. The file is empty | Arrival transaction | Simulation ends.  Final Statistics:  Total number of customers processed: 0  Average amount of time spent waiting: -nan |
| 1. Descendant ordered arrivals time | Arrival transaction  10 8  7 5  6 1  2 2 | Processing an arrival event at time: 2  Processing a departure event at time: 4  Processing an arrival event at time: 6  Processing an arrival event at time: 7  Processing a departure event at time: 7  Processing an arrival event at time: 10  Processing a departure event at time: 12  Processing a departure event at time: 18  Simulation ends.  Final Statistics:  Total number of customers processed: 4  Average amount of time spent waiting: 4 |

# Summary

This project was successfully implemented in our local computer using VSCode and then transferred in the centOS. We have registered no failures while compiling this program.

The program takes a file as an input and processes the arrival and transaction time. The program displays the departure time after processing the file and also counts the total number of customers. At the end, it also displays the average waiting time of each customer.