**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ   
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

Микропроект №2 по дисциплине «Архитектура вычислительных систем»

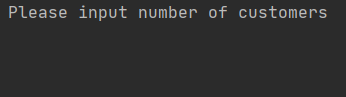
Ибрагимов Константин Леонидович, БПИ192

Задача

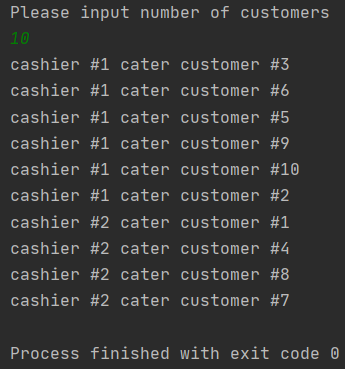
Задача о супермаркете. В супермаркете работают два кассира, покупатели заходят в супермаркет, делают покупки и становятся в очередь к случайному кассиру. Пока очередь пуста, кассир спит, как только появляется покупатель, кассир просыпается. Покупатель спит в очереди, пока не 6 подойдет к кассиру. Создать многопоточное приложение, моделирующее рабочий день супермаркета.

Реализация

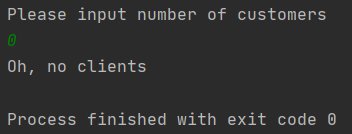
В начале работы программы необходимо ввести количество клиентов 0 <= n <= 100, где n – количество покупателей.



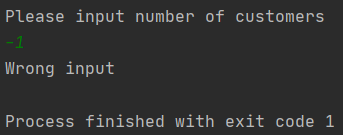
Если вывод был осуществлен верно, программа выводит информацию о том, какой кассир обслужил определенного покупателя.



Также мы не можем исключать ситуацию, что в супермаркете нет посетителей. В такой случае выводится следующее сообщение:



Если же было введено отрицательное число:



Программа писалась с использованием компилятора MinGW.

Текст программы

#include <iostream>  
#include <thread>  
#include <vector>  
#include <queue>  
#include <mutex>  
#include "semaphore.h"  
#include <random>  
  
// Микропроект №2  
// Ибрагимов Константин  
// БПИ192  
  
using namespace std;  
  
// Первая очередь  
sem\_t cashierFirst;  
queue<sem\_t \*> queueFirst;  
mutex mutexQueueFirst;  
  
// Вторая очередь  
sem\_t cashierSecond;  
queue<sem\_t \*> queueSecond;  
mutex mutexQueueSecond;  
  
// Максимальное число очередей на вывод информации  
int numberOfClients;  
mutex numberOfClientsMutex;  
mutex mutexOutput;  
  
static mt19937 mt\_rand(chrono::high\_resolution\_clock::now().time\_since\_epoch().count());  
  
// Рандомный выбор между очередями.  
int randQueue() {  
 return mt\_rand() % 2;  
}  
  
// Симуляция нахождения в очереди.  
void queueProcess(int client\_num) {  
 sem\_t \*semaphore = new sem\_t;  
 int numberOfQueue = randQueue();  
 sem\_init(semaphore, 0, 0);  
 if (numberOfQueue == 0) {  
 mutexQueueFirst.lock();  
 queueFirst.push(semaphore);  
 sem\_post(&cashierFirst);  
 mutexQueueFirst.unlock();  
 } else {  
 mutexQueueSecond.lock();  
 queueSecond.push(semaphore);  
 sem\_post(&cashierSecond);  
 mutexQueueSecond.unlock();  
 }  
  
 sem\_wait(semaphore);  
 mutexOutput.lock();  
 cout << "cashier #" << numberOfQueue + 1 << " cater customer #" << client\_num + 1 << endl;  
 mutexOutput.unlock();  
 numberOfClientsMutex.lock();  
 numberOfClients--;  
 numberOfClientsMutex.unlock();  
}  
  
// Симуляция первого кассира.  
void cashierFirstProcess() {  
 while (!queueFirst.empty()) {  
 sem\_wait(&cashierFirst);  
 mutexQueueFirst.lock();  
 sem\_post(queueFirst.front());  
 queueFirst.pop();  
 mutexQueueFirst.unlock();  
 }  
 sem\_post(&cashierSecond);  
}  
  
// Симуляция работы второго кассира.  
void cashierSecondProcess() {  
 while (!queueSecond.empty()) {  
 sem\_wait(&cashierSecond);  
 mutexQueueSecond.lock();  
 sem\_post(queueSecond.front());  
 queueSecond.pop();  
 mutexQueueSecond.unlock();  
 }  
 sem\_post(&cashierFirst);  
}  
  
// Проверка ввода  
void checkInput() {  
 if (numberOfClients == 0) {  
 cout << "Oh, no clients" << endl;  
 exit(EXIT\_SUCCESS);  
 } else if (numberOfClients < 0) {  
 cout << "Wrong input" << endl;  
 exit(EXIT\_FAILURE);  
 } else if (numberOfClients > 100){  
 cout << "We have no space for these customers";  
 exit(EXIT\_FAILURE);  
 }  
}  
  
int main() {  
 srand(time(nullptr));  
 cout << "Please input number of customers " << endl;  
 cin >> numberOfClients;  
  
 checkInput();  
  
 sem\_init(&cashierFirst, 0, 0);  
 sem\_init(&cashierSecond, 0, 0);  
  
 vector<thread> threads;  
  
 for (int i = 0; i < numberOfClients; ++i)  
 threads.emplace\_back(queueProcess, i);  
  
 threads.emplace\_back(cashierFirstProcess);  
 threads.emplace\_back(cashierSecondProcess);  
  
 for (auto &thread : threads)  
 thread.join();  
}