Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Национальный исследовательский университет

“Высшая школа экономики”»

###### **Факультет компьютерных наук**

Программная инженерия

Исполнитель: Мурзабеков Султан

**МИКРОПРОЕКТ №2**

Работа студента 2 курса бакалавариата группы БПИ-195

по предмету «Архитектура вычислительных систем»

Преподаватель:

Доктор технических наук,

Профессор

Легалов А. И

**Москва 2020**

**Задание:**

Задача о магазине - 2 (забывчивые покупатели). В магазине работают два отдела, каждый отдел обладает уникальным ассортиментом. В каждом отделе работает один продавец. В магазин ходят исключительно забывчивые покупатели, поэтому каждый покупатель носит с собой список товаров, которые желает купить. Покупатель приобретает товары точно в том порядке, в каком они записаны в его списке. Продавец может обслужить только одного покупателя за раз. Покупатель, вставший в очередь, засыпает пока не дойдет до продавца. Продавец засыпает, если в его отделе нет покупателей, и просыпается, если появится хотя бы один. Создать многопоточное приложение, моделирующее работу магазина.

**Решение.**

#include <windows.h>

#include <iostream>

#include <vector>

#include <queue>

#include <conio.h>

#define MAXASSORT 10

using namespace std;

struct Pokupatel

{

int id; //номер покупателя

HANDLE pokupka; //событие что покупка совершена

vector<int> list; //список покупок

};

struct Otdel

{

int id; //название отдела

vector<int> assort; //ассортимент товаров

queue<Pokupatel\*> q; //очередь покупателей

HANDLE wakeup; //событие проснуться продавцу

HANDLE och; //мьютекс доспупа к очереди

};

//проверка есть ли число в векторе

bool is\_in\_vector(vector<int> &v, int val)

{

for (int i = 0; i < v.size(); i++)

{

if (v[i] == val)

return true;

}

return false;

}

Otdel a, b;

HANDLE display,otpusk;

//Поток, моделирующий поведение продавца

DWORD WINAPI Prodavec(PVOID param)

{

Otdel \*o = (Otdel \*)param;

while (true)

{

// WaitForSingleObject(o->wakeup, INFINITE);

WaitForSingleObject(o->och, INFINITE); //синхронизируем доступ к очереди

WaitForSingleObject(otpusk, INFINITE); //Отпускать товар в одно и то же время может только один продавец

int sz = o->q.size();

if (sz) //если очередт не пуста

{

WaitForSingleObject(display, INFINITE);

cout << "Продавец отдела " << o->id << " отпускает товар покупателю " << o->q.front()->id << endl;

ReleaseMutex(display);

SetEvent(o->q.front()->pokupka); //синализировать покупателю что товар отпущен

o->q.pop(); //удаляет покупателя из очереди

}

ReleaseMutex(otpusk); //освободить мьютексы

ReleaseMutex(o->och);

Sleep(1000);

}

return 0;

}

//Поток, моделирующий поведение покупателя

DWORD WINAPI Pokup(PVOID param)

{

Pokupatel \*p = (Pokupatel \*)param;

int i = 0;

//Идем по списку покупок

while (i < p->list.size())

{

//пока очередной товар по списку есть в отделе а

while (i < p->list.size() && is\_in\_vector(a.assort, p->list[i]))// a->check\_assort(buy->list[i]))

{

//становися в очередь в этот отдел

WaitForSingleObject(a.och, INFINITE);

a.q.push(p);

ReleaseMutex(a.och);

WaitForSingleObject(display, INFINITE);

cout << "Покупатель " << p->id << " стал в очередь в отдел 1 за товаром " << p->list[i] << endl;

ReleaseMutex(display);

// SetEvent(a.wakeup);

i++;

}

//пока очередной товар по списку есть в отделе b

while (i < p->list.size() && is\_in\_vector(b.assort, p->list[i]))

{

//становися в очередь в этот отдел

WaitForSingleObject(b.och, INFINITE);

b.q.push(p);

ReleaseMutex(b.och);

WaitForSingleObject(display, INFINITE);

cout << "Покупатель " << p->id << " стал в очередь в отдел 2 за товаром " << p->list[i] << endl;

ReleaseMutex(display);

// SetEvent(b.wakeup);

i++;

}

}

//стоим в очередях и забираем товары

for (int i = 0; i < p->list.size(); i++)

{

WaitForSingleObject(p->pokupka, INFINITE); //ожидаем очередную покупку

WaitForSingleObject(display, INFINITE);

cout << "Покупатель " << p->id << " получил товар " << p->list[i] << endl;

ReleaseMutex(display);

}

//все купили

WaitForSingleObject(display, INFINITE);

cout << "Покупатель " << p->id << " полностью скупился и уходит" << endl;

ReleaseMutex(display);

delete p;

return 0;

}

int main()

{

DWORD id;

setlocale(LC\_ALL, "Russian");//русская локаль

display = CreateMutex(NULL, FALSE, NULL); //мьютекс доступа к дисплею

otpusk = CreateMutex(NULL, FALSE, NULL); //мьютекс отпуска клиенту товара

a.och = CreateMutex(NULL, FALSE, NULL); //мьютексы доступа к очередям отделов

b.och = CreateMutex(NULL, FALSE, NULL);

a.wakeup = CreateEvent(NULL, FALSE, FALSE, NULL); //события пробуждения продавцов

b.wakeup = CreateEvent(NULL, FALSE, FALSE, NULL);

a.id = 1;

//создание ассортимента для первого отдела

cout << "Отдел 1 имеет ассортимент товаров:";

for (int i = 0; i < MAXASSORT / 2; i++)

{

int z;

do

{

z = rand() % MAXASSORT;

} while (is\_in\_vector(a.assort, z)); //берем только уникальные наименования товаров

a.assort.push\_back(z); //добавляем их

cout << " " << a.assort.back();

}

cout << endl;

CreateThread(NULL, 0, Prodavec, (LPVOID)&a, 0, &id); //создать поток продавка 1 отдела

b.id = 2;

cout << "Отдел 2 имеет ассортимент товаров:";

//создание ассортимента для второго отдела

for (int i = 0; i < MAXASSORT / 2; i++)

{

int z;

do

{

z = rand() % MAXASSORT;

} while (is\_in\_vector(a.assort, z) || is\_in\_vector(b.assort, z)); //берем только уникальные наименования товаров и чтобы их не было в первом отделе

b.assort.push\_back(z); //добавляем их

cout << " " << b.assort.back();

}

cout << endl;

CreateThread(NULL, 0, Prodavec, (LPVOID)&b, 0, &id); //создать поток продавка 2 отдела

cout << "Esc - выход, другие клавиши запускают покупателя в магазин" << endl;

int npookup = 0;

Pokupatel \*p;

srand(GetTickCount());

while (\_getch() != 27) //ожидание нажатия клавиши

{

p = new Pokupatel();

p->id = ++npookup;

int n = rand() % 5 + 5; //размер списка покупок

//выводим список покупок

WaitForSingleObject(display, INFINITE);

cout << "Создан покупатель " << p->id << " со списком покупок:";

for (int i = 0; i < n; i++) //Генерируем список покупок

{

p->list.push\_back(rand() % MAXASSORT);

cout << " " << p->list.back();

}

cout << endl;

ReleaseMutex(display);

p->pokupka = CreateEvent(NULL, FALSE, FALSE, NULL);

CreateThread(NULL, 0, Pokup, (LPVOID)p, 0, &id);

}

return 0;

Результат выполнения программы приведен на рисунке 1.

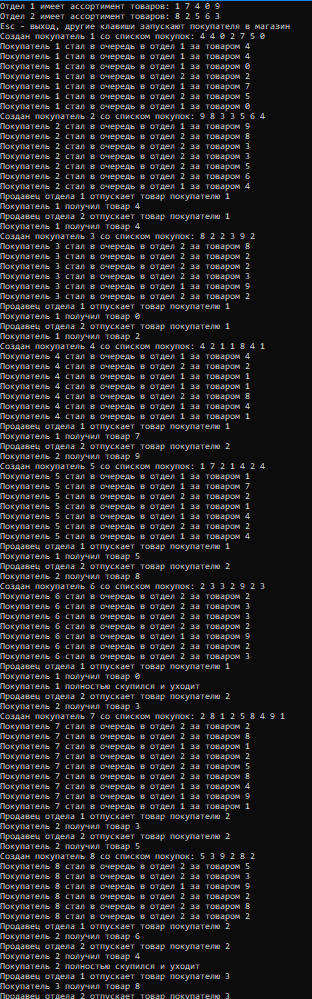


Рисунок 1 ­– Результат выполнения программы

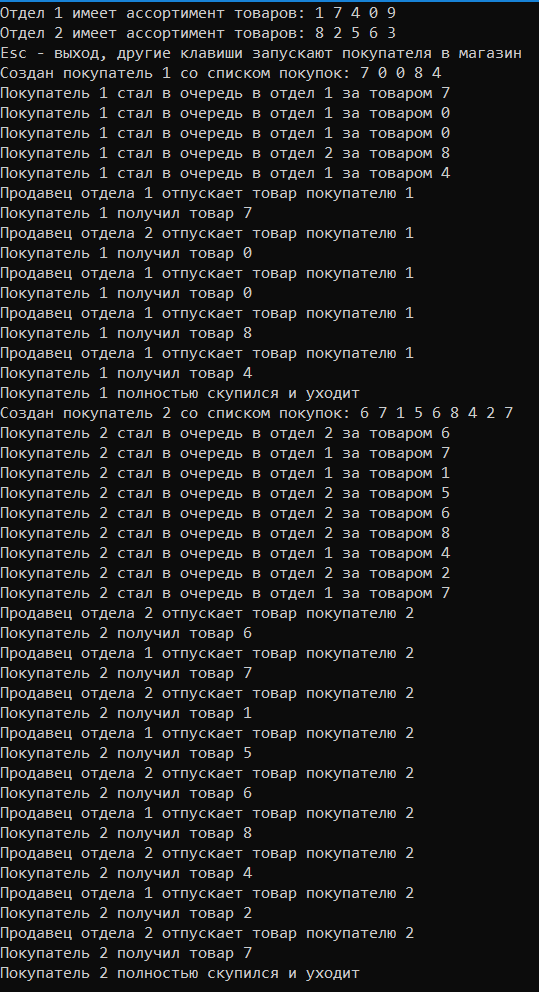


Рисунок 2 ­– Тест №2

**Список использованных источников**

1 <https://math.semestr.ru/math/taylor.php>

2 <https://ru.wikipedia.org/wiki/Логарифм>

3 <https://habr.com/ru/post/182610/>

4 https://www.geeksforgeeks.org/multithreading-in-cpp/