МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

ОТЧЁТ

по лабораторной работе №7

“Синхронизация потоков.”

Выполнил:
студент группы ПО-9
Солышко Д.А.

Проверил:
Козик И.Д.

Брест 2023

Лабораторная работа №7
Вариант №19

Цель работы: научиться синхронизировать работу с потоками в приложениях.

Основное содержание работы.

Необходимо написать программу, используя знания, полученные в лабораторной работе №5. Потоки должны быть синхронизированы для выполнения поставленной задачи или код должен выполнять задачи последовательно.

Задание: 3. Создать симулятор рынка с минимум 3 торговцами. Торговцы обслуживают по одному человеку в очереди, а после того, как очередь закончилась – начинаю обслуживать людей в очередях других торговцев.

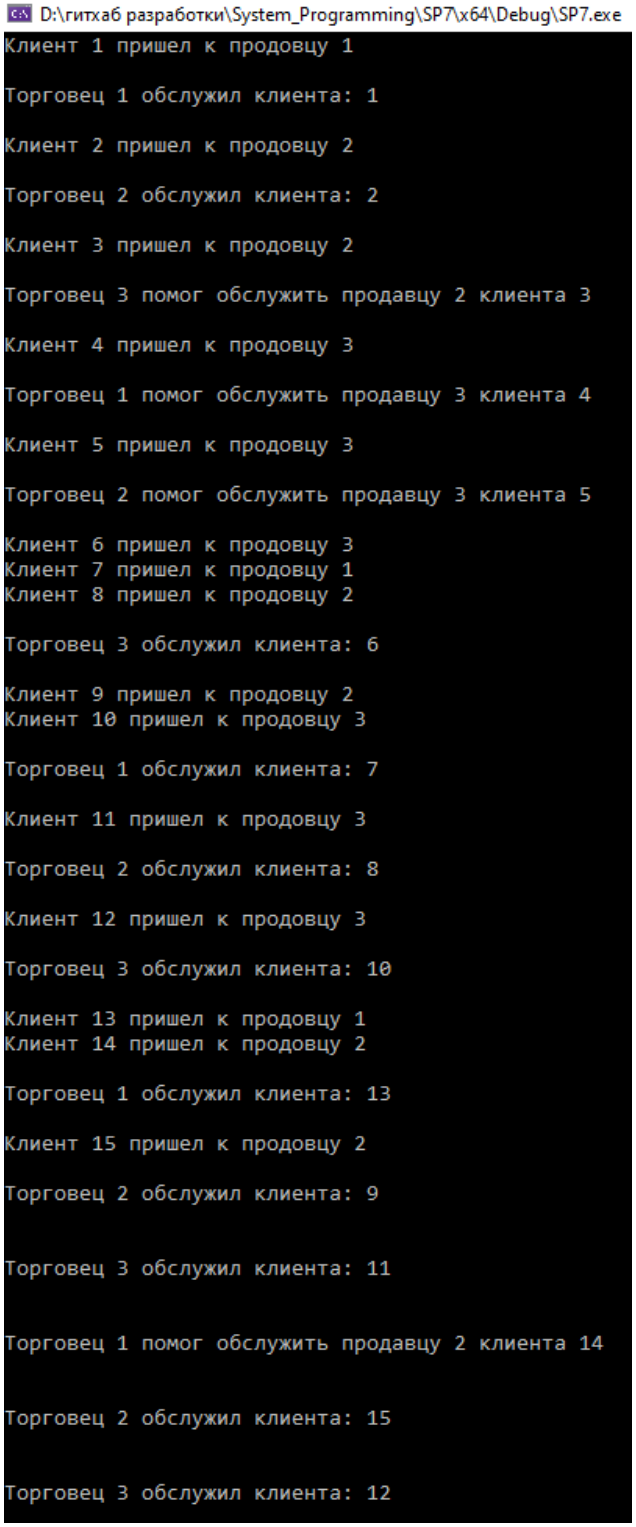
Текст программы:

```
#include <iostream>
#include <thread>
#include <mutex>
#include <queue>
#include <windows.h>

using namespace std;

class Clients {
private:
   queue<int> q;
   mutex mtx;

public:
   void enqueue(int customer) {
      mtx.lock();
      q.push(customer);
      mtx.unlock();
   }

   int dequeue() {
      int customer = -1;
      mtx.lock();
      if (!q.empty()) {
         customer = q.front();
         q.pop();
      }
      mtx.unlock();
      return customer;
   }
};

class Trader {
private:
   Clients& ownQueue;
   Clients& sharedQueue1;
   Clients& sharedQueue2;

public:
   int traderID;
   Trader(Clients& ownQ, Clients& sharedQ1, Clients& sharedQ2, int id) : ownQueue(ownQ),
sharedQueue1(sharedQ1), sharedQueue2(sharedQ2), traderID(id) {}
```

```
    void serveCustomers(int traderID2, int traderID3) {
        while (true) {
            int customer = ownQueue.dequeue();
            if (customer == -1) {
                customer = sharedQueue1.dequeue();
                if (customer != -1) {
                    cout << "\nТорговец " << traderID << " помог обслужить продавцу " << traderID2 << " клиента "
<< customer << endl << endl;
                    Sleep(3000);
                }
                else {
                    customer = sharedQueue2.dequeue();
                    if (customer != -1) {
                        cout << "\nТорговец " << traderID << " помог обслужить продавцу " << traderID3 << " клиента "
<< customer << endl << endl;
                        Sleep(3000);
                    }
                }
                continue;
            }
            cout << "\nТорговец " << traderID << " обслужил клиента: " << customer << endl << endl;
            Sleep(2000);
        }
    }
};

int main() {
    setlocale(LC_ALL, "ru");
    Clients marketclients1, marketclients2, marketclients3;

    Trader trader1(marketclients1, marketclients2, marketclients3, 1);
    Trader trader2(marketclients2, marketclients1, marketclients3, 2);
    Trader trader3(marketclients3, marketclients2, marketclients1, 3);

    thread thread1([&]() {
        trader1.serveCustomers(trader2.traderID, trader3.traderID);
        });

    thread thread2([&]() {
        trader2.serveCustomers(trader1.traderID, trader3.traderID);
        });

    thread thread3([&]() {
        trader3.serveCustomers(trader2.traderID, trader1.traderID);
        });

    for (int i = 1; i <= 15; ++i) {
        if (i % 6 == 1)
        {
            cout << "Клиент " << i << " пришел к продовцу " << trader1.traderID << endl;
            marketclients1.enqueue(i);
        }
        else if (i % 6 == 2 || i % 6 == 3)
        {
            cout << "Клиент " << i << " пришел к продовцу " << trader2.traderID << endl;
            marketclients2.enqueue(i);
        }
        else
        {
            cout << "Клиент " << i << " пришел к продовцу " << trader3.traderID << endl;
            marketclients3.enqueue(i);
        }
        Sleep(500);
```

```
    }

    thread1.join();
    thread2.join();
    thread3.join();

    return 0;
}
```

Результаты программы:



Вывод: я научился синхронизировать работу с потоками в приложениях.