

# **Правительство Российской Федерации**

Федеральное государственное автономное образовательное учреждение  
высшего образования «Национальный исследовательский университет  
«Высшая школа экономики»

Факультет компьютерных наук

## **Пояснительная записка к домашнему заданию №4**

**По дисциплине**

**“Архитектура вычислительных систем”**

**На тему**

**“ Разработка многопоточных приложений с использованием  
OpenMP”**

Работу выполнил

Студент группы БПИ-194

\_\_\_\_\_

М. Г. Савинов

подпись, дата

Работу проверил

\_\_\_\_\_

А.И. Легалов

подпись, дата

Москва 2020

## Содержание

Постановка задачи.....	3
Описание задачи.....	3
Пояснение задачи .....	4
Пример работы программы .....	5
Приложение .....	9
Список использованной литературы .....	13

## Постановка задачи

Первая военная задача. Темной-темной ночью прапорщики Иванов, Петров и Нечепорчук занимаются хищением военного имущества со склада родной военной части. Будучи умными людьми и отличниками боевой и строевой подготовки, прапорщики ввели разделение труда: Иванов выносит имущество со склада, Петров грузит его в грузовик, а Нечепорчук подсчитывает рыночную стоимость добычи. Требуется составить многопоточное приложение, моделирующее деятельность прапорщиков. При решении использовать парадигму «производитель-потребитель».

## Описание задачи

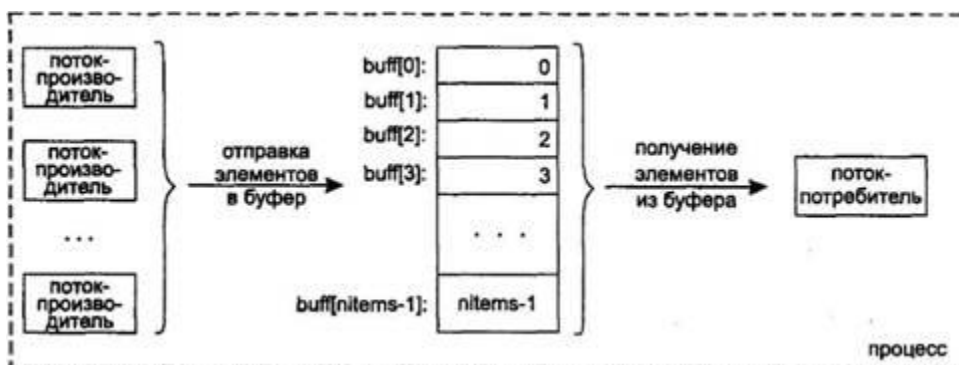
- 1) Изучить применение OpenMP для разработки многопоточных приложений.
  - a. Для заданной программы была использована стандартная библиотека C++, библиотека OpenMP для запуска потоков и парадигма «производитель-потребитель».
  - b. Создадим 3 класса (1 производитель и 2 потребителя).
  - c. Класс производитель `ivanov` берет со склада имущество по одной штуке и передает производителю `petrov` (загружает в очередь).
  - d. Класс потребитель `petrov` отгружает полученный груз в грузовик и передает производителю `necheporchuk`.
  - e. Класс-потребитель `necheporchuk` добавляет стоимость полученного груза в общую прибыль и вычеркивает груз из списка (удаляет из очереди).
  - f. Для синхронизации потоков используется метод `sleep_for`, для удобства использования программы для класса `ivanov` выставлено значение 5000ms, для `petrov` 3000ms, а для `necheporchuk` 2000ms, так как потоки работают параллельно, то пока потоки потребителей производят действия, поток производителя добавит новый элемент в очередь.
  - g. Для взаимно-исключающего выполнения потоков используется `mutex`, а именно его методы `lock` и `unlock`.
  - h. Генерация количества и стоимостей грузов выполняется с помощью использования заголовочного файла `gandom` на основе текущего времени. Для удобства использования программы выставлен диапазон целых чисел от 5 до 30, однако его можно менять, что не повлияет на работоспособность программы.
  - i. Потоки вызываются с помощью библиотеки OpenMP, а именно разделение на секции, которые определяет разделы кода, которые должны быть разделены между всеми потоками.
- 2) Для выполнения программы не требуется входных данных, однако в коде программы можно изменить границы генерации чисел.
- 3) Программа последовательно выдает действия всех трех потоков и успешно завершается с кодом 0.

## Пояснение задачи

Как уже было сказано выше для решения задачи использована парадигма «производитель-потребитель».

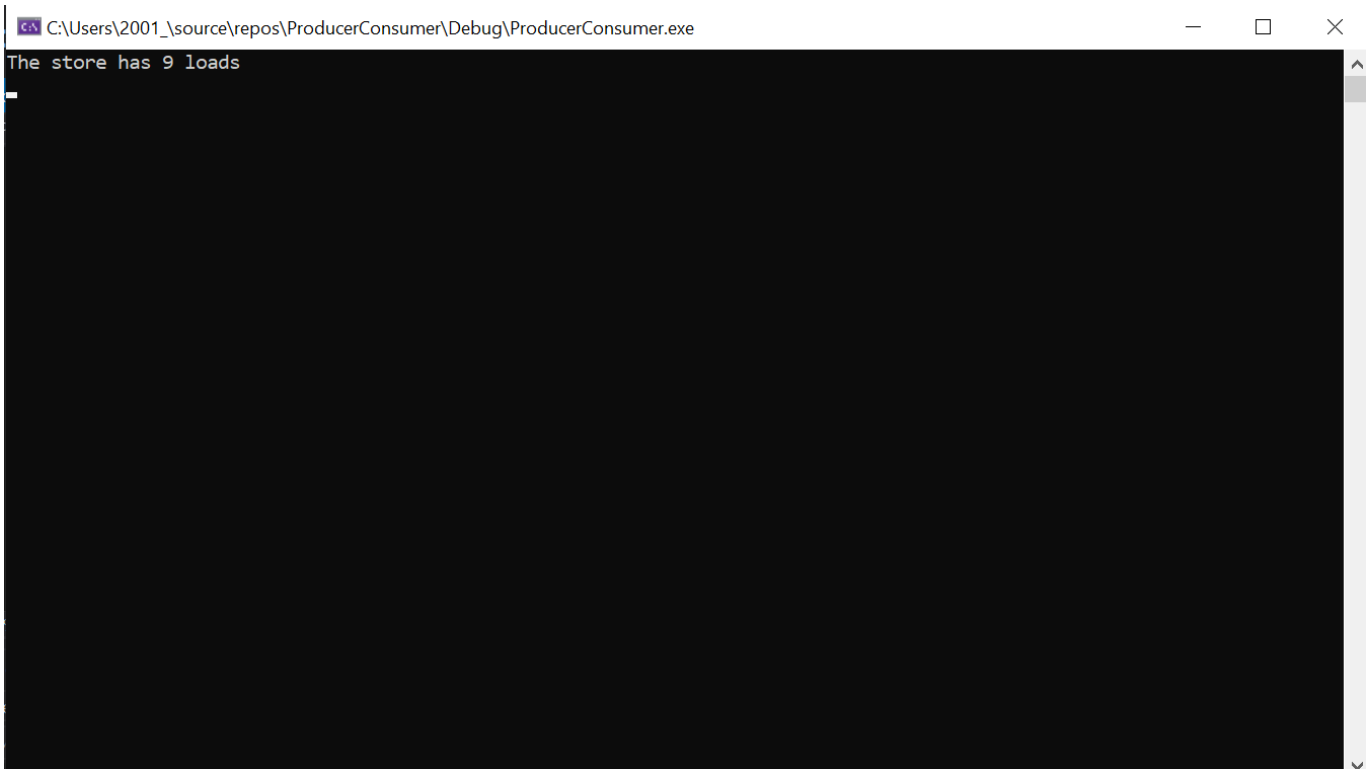
Производители и потребители – это парадигма взаимодействующих неравноправных потоков. Одни потоки «производят» данные, другие их «потребляют». Часто такие потоки организуются в конвейер, через который проходит информация. Каждый поток конвейера потребляет выход своего предшественника и производит входные данные для своего последователя. Другой распространенный способ организации потоков – древовидная структура или сети слияния, на этом основан, в частности, метод дихотомии.

Один или несколько производителей (потоков или процессов) создают данные, которые обрабатываются одним или несколькими потребителями. Эти данные передаются между производителями и потребителями с помощью одной из форм IPC.



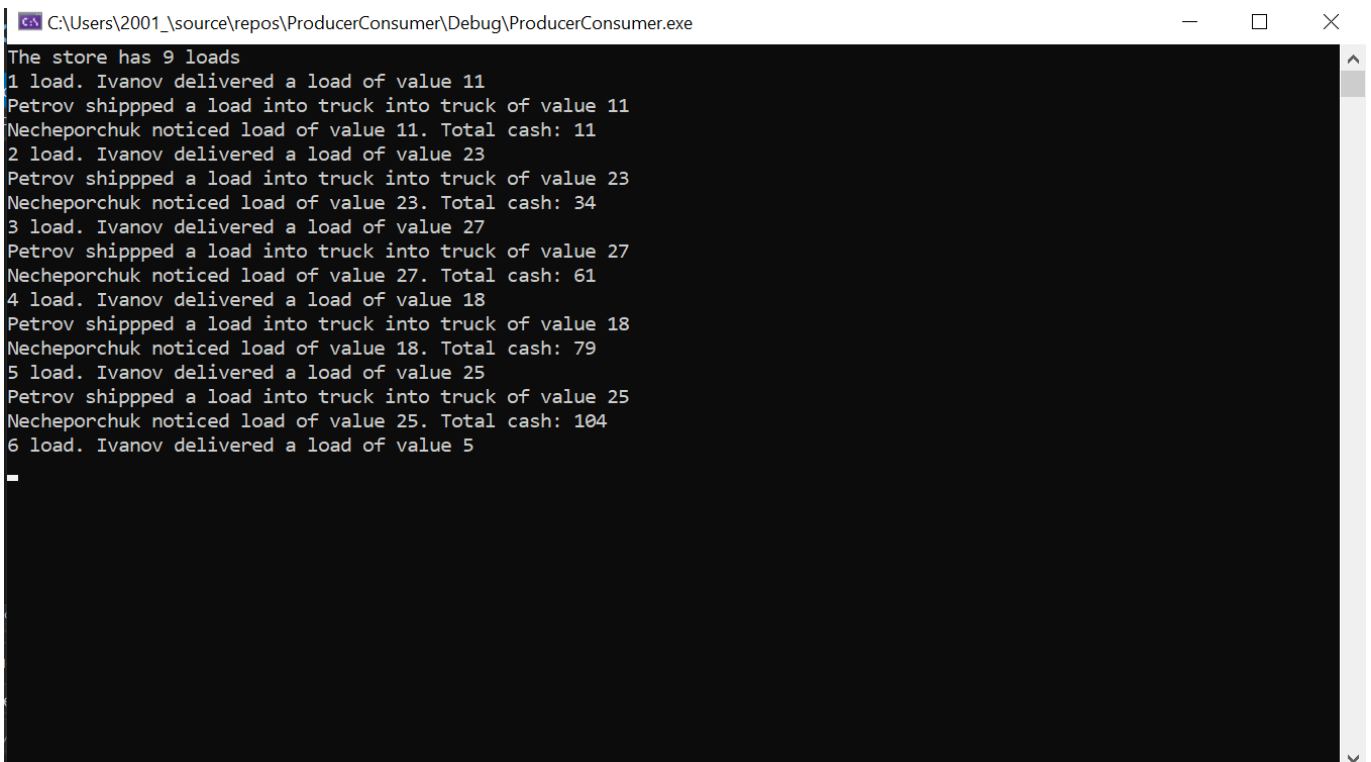
## Пример работы программы

1) Сгенерированное число груза на складе – 9, ждем пока иванов принесет первый груз.



```
C:\Users\2001_\source\repos\ProducerConsumer\Debug\ProducerConsumer.exe
The store has 9 loads
```

Дошли до 6 груза, заметим, что действие иванов мгновенное, с учетом того, что несет груз он 5000ms, это показывает нам то, что потоки работают параллельно. На данном скриншоте ждем, пока petrov положит его в грузовик.



```
C:\Users\2001_\source\repos\ProducerConsumer\Debug\ProducerConsumer.exe
The store has 9 loads
1 load. Ivanov delivered a load of value 11
Petrov shipped a load into truck into truck of value 11
Necheporchuk noticed load of value 11. Total cash: 11
2 load. Ivanov delivered a load of value 23
Petrov shipped a load into truck into truck of value 23
Necheporchuk noticed load of value 23. Total cash: 34
3 load. Ivanov delivered a load of value 27
Petrov shipped a load into truck into truck of value 27
Necheporchuk noticed load of value 27. Total cash: 61
4 load. Ivanov delivered a load of value 18
Petrov shipped a load into truck into truck of value 18
Necheporchuk noticed load of value 18. Total cash: 79
5 load. Ivanov delivered a load of value 25
Petrov shipped a load into truck into truck of value 25
Necheporchuk noticed load of value 25. Total cash: 104
6 load. Ivanov delivered a load of value 5
```

Программа успешно завершилось с кодом 0, necheporchuk подсчитал общую ценность грузов.

```
Консоль отладки Microsoft Visual Studio

2 load. Ivanov delivered a load of value 23
Petrov shipped a load into truck into truck of value 23
Necheporchuk noticed load of value 23. Total cash: 34
3 load. Ivanov delivered a load of value 27
Petrov shipped a load into truck into truck of value 27
Necheporchuk noticed load of value 27. Total cash: 61
4 load. Ivanov delivered a load of value 18
Petrov shipped a load into truck into truck of value 18
Necheporchuk noticed load of value 18. Total cash: 79
5 load. Ivanov delivered a load of value 25
Petrov shipped a load into truck into truck of value 25
Necheporchuk noticed load of value 25. Total cash: 104
6 load. Ivanov delivered a load of value 5
Petrov shipped a load into truck into truck of value 5
Necheporchuk noticed load of value 5. Total cash: 109
7 load. Ivanov delivered a load of value 19
Petrov shipped a load into truck into truck of value 19
Necheporchuk noticed load of value 19. Total cash: 128
8 load. Ivanov delivered a load of value 24
Petrov shipped a load into truck into truck of value 24
Necheporchuk noticed load of value 24. Total cash: 152
9 load. Ivanov delivered a load of value 24
Petrov shipped a load into truck into truck of value 24
Necheporchuk noticed load of value 24. Total cash: 176

C:\Users\2001_\source\repos\ProducerConsumer\Debug\ProducerConsumer.exe (процесс 10984) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

2)Посмотрим на аналогичные пример, теперь было сгенерировано 18 грузов.

```
C:\Users\2001_\source\repos\ProducerConsumer\Debug\ProducerConsumer.exe

The store has 18 loads
```

Программа также успешно завершилось с кодом 0.

```
Консоль отладки Microsoft Visual Studio

11 load. Ivanov delivered a load of value 24
Petrov shipped a load into truck into truck of value 24
Necheporchuk noticed load of value 24. Total cash: 166
12 load. Ivanov delivered a load of value 25
Petrov shipped a load into truck into truck of value 25
Necheporchuk noticed load of value 25. Total cash: 191
13 load. Ivanov delivered a load of value 27
Petrov shipped a load into truck into truck of value 27
Necheporchuk noticed load of value 27. Total cash: 218
14 load. Ivanov delivered a load of value 24
Petrov shipped a load into truck into truck of value 24
Necheporchuk noticed load of value 24. Total cash: 242
15 load. Ivanov delivered a load of value 24
Petrov shipped a load into truck into truck of value 24
Necheporchuk noticed load of value 24. Total cash: 266
16 load. Ivanov delivered a load of value 15
Petrov shipped a load into truck into truck of value 15
Necheporchuk noticed load of value 15. Total cash: 281
17 load. Ivanov delivered a load of value 24
Petrov shipped a load into truck into truck of value 24
Necheporchuk noticed load of value 24. Total cash: 305
18 load. Ivanov delivered a load of value 14
Petrov shipped a load into truck into truck of value 14
Necheporchuk noticed load of value 14. Total cash: 319

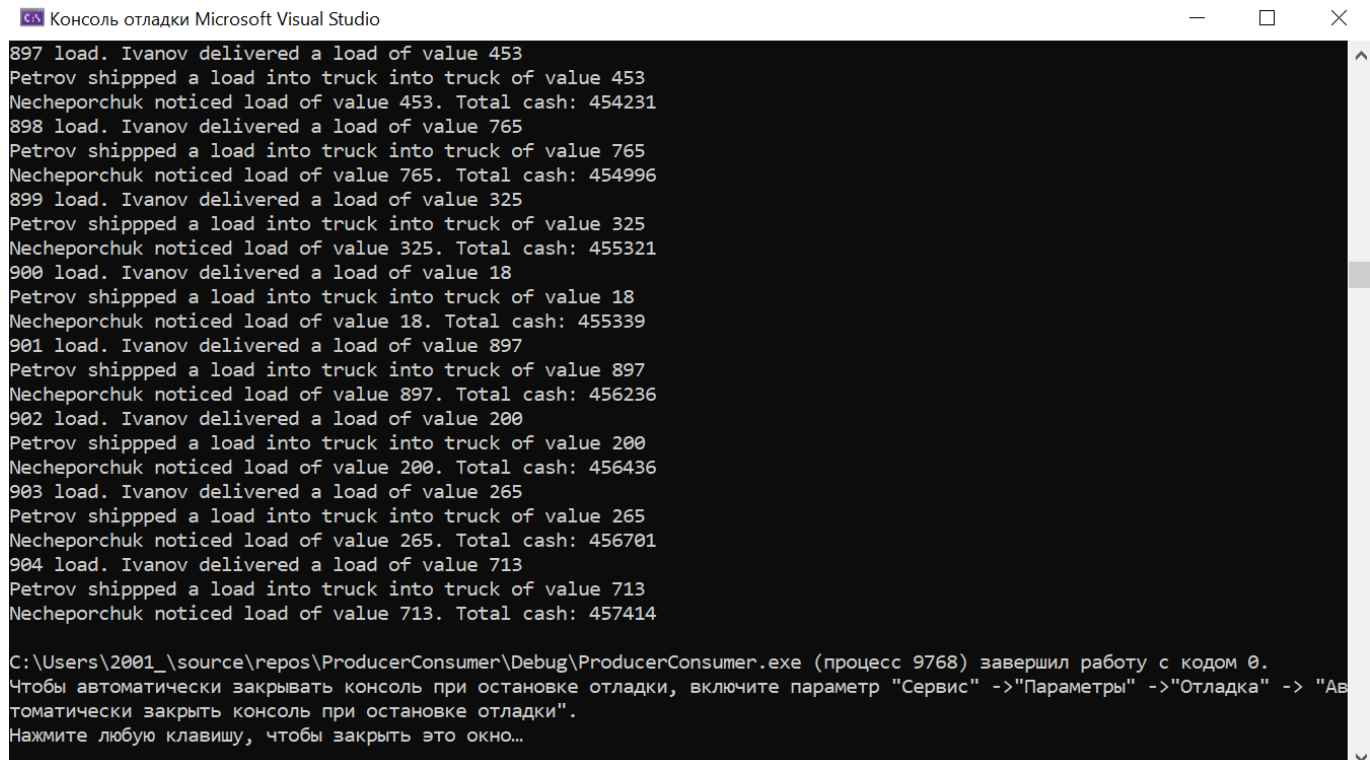
C:\Users\2001_\source\repos\ProducerConsumer\Debug\ProducerConsumer.exe (процесс 22520) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

3)Давайте изменим границы random от 5 до 1000, соответственно изменим и время ожидания до 10ms у каждого класса, чтобы программа отработала быстрее. Было сгенерировано 904 груза.

```
C:\Users\2001_\source\repos\ProducerConsumer\Debug\ProducerConsumer.exe

The store has 904 loads
```

Программа также успешно исполнилась и завершилась с кодом 0.



```
Консоль отладки Microsoft Visual Studio

897 load. Ivanov delivered a load of value 453
Petrov shipped a load into truck into truck of value 453
Necheporchuk noticed load of value 453. Total cash: 454231
898 load. Ivanov delivered a load of value 765
Petrov shipped a load into truck into truck of value 765
Necheporchuk noticed load of value 765. Total cash: 454996
899 load. Ivanov delivered a load of value 325
Petrov shipped a load into truck into truck of value 325
Necheporchuk noticed load of value 325. Total cash: 455321
900 load. Ivanov delivered a load of value 18
Petrov shipped a load into truck into truck of value 18
Necheporchuk noticed load of value 18. Total cash: 455339
901 load. Ivanov delivered a load of value 897
Petrov shipped a load into truck into truck of value 897
Necheporchuk noticed load of value 897. Total cash: 456236
902 load. Ivanov delivered a load of value 200
Petrov shipped a load into truck into truck of value 200
Necheporchuk noticed load of value 200. Total cash: 456436
903 load. Ivanov delivered a load of value 265
Petrov shipped a load into truck into truck of value 265
Necheporchuk noticed load of value 265. Total cash: 456701
904 load. Ivanov delivered a load of value 713
Petrov shipped a load into truck into truck of value 713
Necheporchuk noticed load of value 713. Total cash: 457414

C:\Users\2001_\source\repos\ProducerConsumer\Debug\ProducerConsumer.exe (процесс 9768) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрывать консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```



## Приложение

### Код программы:

```
#include <iostream>

#include <queue>

#include <mutex>

#include <random>

#include <future>

#include <omp.h>

using namespace std;

mutex m;

//переменная для контроля потоков petrov и pecheporchuk

int semaphore = 1;

//очередь для подсчета груза

queue<int> taken_property;

//количество добычи

int loads = 0;

//генератор случайных чисел

mt19937 gen((int)time(0));

uniform_int_distribution<int> uid(5, 30);

int max_count = uid(gen);

//class producer

class Ivanov {

public:

    explicit Ivanov() {}

    void action() {

        //симуляции кражи первого груза

        this_thread::sleep_for(5000ms);

        while (true) {
```

```

        m.lock();

        //ценность груза
        int elem = uid(gen);

        //добавляем в очередь
        taken_property.push(elem);

        ++loads;

        cout << loads << " load. Ivanov delivered a load of value " << elem << "\n";

        m.unlock();

        //отправляем ivanov за следующим грузом
        this_thread::sleep_for(5000ms);

        //если склад опустел
        if (loads == max_count)break;

        //заставляем ждать, если товарищи еще не доделали свои дела
        while (!taken_property.empty()) this_thread::sleep_for(10ms);
    }

}

};

//class consumer petrov
class Petrov {
public:
    explicit Petrov() {}

    void action() {
        while (true) {
            m.lock();

            //если ivanov принес груз
            if (!taken_property.empty()) {
                //время на погрузку в грузовик
                this_thread::sleep_for(3000ms);

                int elem = taken_property.front();

                cout << "Petrov shipped a load into truck of value " << elem << "\n";

                m.unlock();

                //уведомляем neshcheporuchuk, что груз доставлен в грузовик

```

```

        --semaphore;
    }

    else m.unlock();

    //если груз закончился завершаем поток
    if (loads == max_count)break;

    //ждем, если necheporchuk еще не подсчитал предыдущий груз
    while (!semaphore) this_thread::sleep_for(10ms);

    }

}

};

//class consumer necheporchuk
class Nечeporchuk {
public:
    explicit Nечeporchuk() {}
    void action() {
        while (true) {
            //ждем, если petrov еще не доставил груз в грузовик
            while (semaphore)this_thread::sleep_for(10ms);

            //время на подсчет суммы
            this_thread::sleep_for(2000ms);

            m.lock();

            int elem = taken_property.front();

            //удаляем груз из очереди
            taken_property.pop();

            total_cash += elem;

            cout << "Necheporchuk noticed load of value " << elem << ". Total cash: " << total_cash <<
            "\n";

            //сообщаем petrov, что готовы считать следующий груз
            ++semaphore;

            m.unlock();

            //завершаем поток, если груз закончился
            if (loads == max_count)break;

```

```

        //если груза еще нету, ждем
        while (taken_property.empty())this_thread::sleep_for(10ms);
    }
}

private:
    long long total_cash = 0;
};

void MultiThreads() {
    Ivanov ivanov;
    Petrov petrov;
    Necheporchuk necheporchuk;
    //используем секции для того, чтобы запустить все 3 потока
#pragma omp parallel sections
    {
#pragma omp section
        ivanov.action();
#pragma omp section
        petrov.action();
#pragma omp section
        necheporchuk.action();
    }
}

int main()
{
    cout << "The store has " << max_count << " loads\n";
    //Вызываем потоки
    MultiThreads();
    return 0;
}

```

## Список использованной литературы

1. ВикиЧтение. Стивенс Уильям Ричард. «UNIX: взаимодействие процессов. 7.3. Схема производитель-потребитель.» (<https://it.wikireading.ru/24842>).
2. Легалов А.И.(2020) «Многопоточное программирование. OpenMP» (<http://www.softcraft.ru/edu/comparch/practice/thread/03-openmp/>)
3. Microsoft docs «Директивы OpenMP» (<https://docs.microsoft.com/ru-ru/cpp/parallel/openmp/reference/openmp-directives?view=msvc-160#sections-openmp>)