

**Правительство Российской Федерации**

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

Факультет компьютерных наук Департамент программной инженерии

**Отчет к микропроекту по дисциплине  
«Архитектура вычислительных систем»**

Работу выполнил:

Студент группы БПИ195 Скарлупин М. О.

Вариант № 26.

**Москва 2020**

## Задание

4. *Задача об обедающих философах.* Пять философов сидят возле круглого стола. Они проводят жизнь, чередуя приемы пищи и размышления. В центре стола находится большое блюдо спагетти. Спагетти длинные и запутанные, философам тяжело управляться с ними, поэтому каждый из них, что бы съесть порцию, должен пользоваться двумя вилами. К несчастью, философам дали только пять вилок. Между каждой парой философов лежит одна вилка, поэтому эти высококультурные и предельно вежливые люди договорились, что каждый будет пользоваться только теми вилами, которые лежат рядом с ним (слева и справа). Написать многопоточную программу, моделирующую поведение философов с помощью семафоров. Программа должна избегать фатальной ситуации, в которой все философы голодны, но ни один из них не может взять обе вилки (например, каждый из философов держит по одной вилки и не хочет отдавать ее). Решение должно быть симметричным, то есть все потоки-философы должны выполнять один и тот же код.

## Решение

Идея состоит в том, чтобы найти решение, когда ни один из философов не голодал, то есть хотя бы когда-нибудь имел возможность взять вилки, необходимые ему для еды.

Определены следующие классы:

`Fork` - представляет собой вилку за столом; единственный член этой структуры - `std::mutex`, который будет заблокирован, когда философ поднимет вилку, и разблокирован, когда он ее положит.

`Table` - представляет собой круглый стол, за которым обедают философы. У него есть массив вилок, а также атомарное логическое значение, которое указывает, что стол готов, чтобы философы начали думать и есть.

`Philosopher` - представляет философа, обедающего за столом. У него есть название и ссылка на вилки слева и справа.

Большая часть реализации решения принадлежит классу `Philosopher`. Когда создается объект этого класса, запускается поток. Этот поток присоединяется, когда объект уничтожается. Поток запускает цикл обдумывания и приема пищи до тех пор, пока обед не будет закончен, установив для элемента `ready` таблицы значение `false`. В классе `Philosopher` есть три основных метода:

`dine()` - функция потока. По сути, представляет из себя цикл обдумывания и приёма пищи.

`think()` - это метод, который представляет собой период размышления. Чтобы смоделировать это, поток спит в течение случайного периода времени.

`eat()` - это метод, моделирующий питание. Левая и правая вилки извлекаются без взаимоблокировок с помощью `std::lock`. После того, как вилки, то есть мьютексы, получены, их владение передается объекту `std::lock_guard`, так, что мьютексы освобождаются при возврате функции.

Теперь используем данные классы и методы:

Мы создаем объект таблицы и массив `Philosopher`. После создания пяти объектов этого класса запускается их собственный рабочий поток. Затем философы соревнуются за вилки (то есть мьютексы), едят и думают, пока обед не будет завершен.

## Тестирование

На рисунках 1,2 и 3 видно, что программа каждый раз выполняется по разному, а значит она выполняется параллельно.

```
Консоль отладки Microsoft Visual Studio

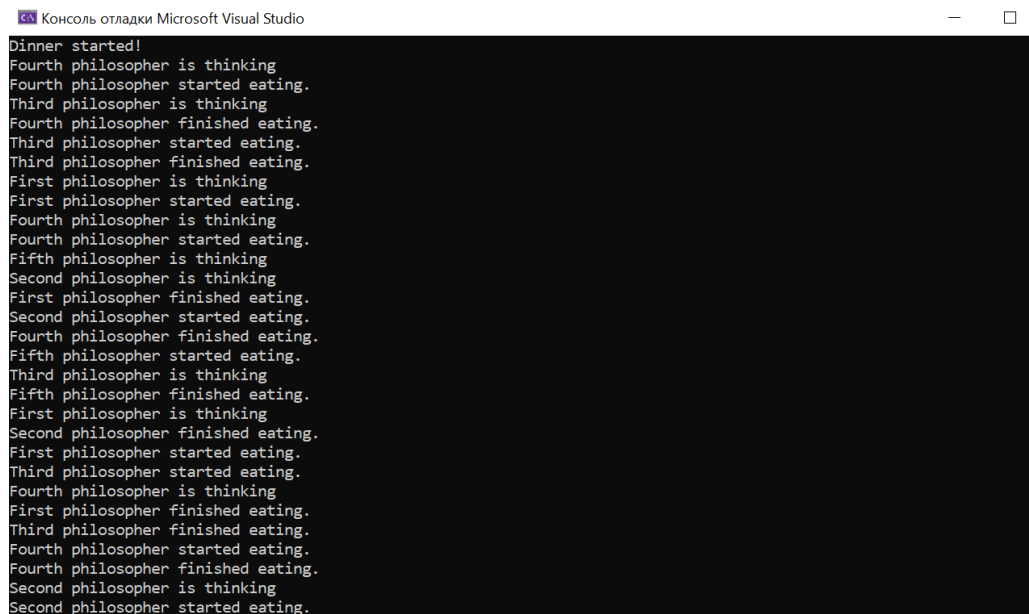
Dinner started!
Third philosopher is thinking
Third philosopher started eating.
Third philosopher finished eating.
Second philosopher is thinking
Fourth philosopher is thinking
Second philosopher started eating.
Fourth philosopher started eating.
Third philosopher is thinking
Fifth philosopher is thinking
First philosopher is thinking
Second philosopher finished eating.
First philosopher started eating.
Fourth philosopher finished eating.
Third philosopher started eating.
First philosopher finished eating.
Fifth philosopher started eating.
Third philosopher finished eating.
Fifth philosopher finished eating.
Second philosopher is thinking
Second philosopher started eating.
Second philosopher finished eating.
First philosopher is thinking
First philosopher started eating.
Fourth philosopher is thinking
Fourth philosopher started eating.
Fifth philosopher is thinking
First philosopher finished eating.
Fourth philosopher finished eating.
Fifth philosopher started eating.
```

Рисунок 1

```
Консоль отладки Microsoft Visual Studio

Dinner started!
Fourth philosopher is thinking
Fifth philosopher is thinking
Fourth philosopher started eating.
Fourth philosopher finished eating.
Fifth philosopher started eating.
Second philosopher is thinking
Second philosopher started eating.
Second philosopher finished eating.
Fifth philosopher finished eating.
Fourth philosopher is thinking
Fourth philosopher started eating.
Third philosopher is thinking
Fourth philosopher finished eating.
Third philosopher started eating.
First philosopher is thinking
First philosopher started eating.
First philosopher finished eating.
First philosopher is thinking
First philosopher started eating.
Third philosopher finished eating.
Fifth philosopher is thinking
First philosopher finished eating.
Fifth philosopher started eating.
Second philosopher is thinking
Second philosopher started eating.
Fourth philosopher is thinking
Fifth philosopher finished eating.
Fourth philosopher started eating.
Fourth philosopher finished eating.
```

Рисунок 2



```
Консоль отладки Microsoft Visual Studio
Dinner started!
Fourth philosopher is thinking
Fourth philosopher started eating.
Third philosopher is thinking
Fourth philosopher finished eating.
Third philosopher started eating.
Third philosopher finished eating.
First philosopher is thinking
First philosopher started eating.
Fourth philosopher is thinking
Fourth philosopher started eating.
Fifth philosopher is thinking
Second philosopher is thinking
First philosopher finished eating.
Second philosopher started eating.
Fourth philosopher finished eating.
Fifth philosopher started eating.
Third philosopher is thinking
Fifth philosopher finished eating.
First philosopher is thinking
Second philosopher finished eating.
First philosopher started eating.
Third philosopher started eating.
Fourth philosopher is thinking
First philosopher finished eating.
Third philosopher finished eating.
Fourth philosopher started eating.
Fourth philosopher finished eating.
Second philosopher is thinking
Second philosopher started eating.
```

Рисунок 3

## Список используемых источников

1. Андреев Н. (2019) «Семафор на событиях C++»

(<https://habr.com/ru/post/476940/>)

Просмотрено 10.12.2020

2. Грегори Р. Эндрюс (2003) «Основы многопоточного, параллельного и распределённого программирования»

(<https://l.wzm.me/coder/custom/parallel.programming/main.htm>)

Просмотрено 10.12.2020

3. Легалов А.И. (2020) «Многопоточное программирование. Синхронизация»

(<http://softcraft.ru/edu/comparch/practice/thread/02-sync/>)

Просмотрено: 13.12.2020