

Национальный исследовательский университет “Высшая Школа Экономики”,  
Факультет компьютерных наук  
Департамент программной инженерии

**«Задача об обедающих философах»**  
Пояснительная записка

Исполнитель:  
Студент группы БПИ199  
Федченко Всеволод Александрович

Москва 2020

# Оглавление

<b>1. Текст задания.....</b>	<b>3</b>
<b>2. Применяемые расчетные методы.....</b>	<b>4</b>
2.1 Запуск программы.....	4
2.2 Алгоритм.....	4
2.2 Входные данные.....	4
2.3 Выходные данные.....	5
<b>3. Тестовые примеры.....</b>	<b>6</b>
<b>4. Список используемых источников.....</b>	<b>7</b>

## 1. Текст задания

*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 вилками. 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 вилками, 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 вилке и не хочет отдавать ее). Решение должно быть симметричным, то есть все потоки-философы должны выполнять один и тот же код. [1][2]

## 2. Применяемые расчетные методы

### 2.1 Запуск программы

Для компиляции и запуска программы в консоль необходимо ввести строчку `c++ ./avs_mp_2.cpp -o main -lpthread -std=c++11 && ./main <argv1>`, где `<argv1>` опциональный аргумент, отвечающий за количество итераций.

### 2.2 Алгоритм

Для того, чтобы избежать ситуации, когда каждый философ (поток) взял по одной вилке и ждет, когда освободится еще одна вилка, и, следовательно, ждет вечно, так как никто не отдаст свою вилку до тех пор, пока не поест (так называемый deadlock[3] потоков) используется решение, впервые предложенное Дейкстрой. Алгоритм представляет собой присвоение частичного порядка ресурсам и установление соглашения, что ресурсы запрашиваются в указанном порядке, а возвращаются в обратном порядке.

Для решения поставленной задачи обозначенным выше алгоритмом, каждой вилке и каждому философу был присвоен номер от 0 до 4. Далее, для каждого сидящего за столом философа была назначена левая и правая вилка таким образом, что левая вилка имеет наименьший номер из тех, что может взять данный философ (ближающие справа и слева), а правая вилка — наибольший. Для философа №4 номер левой вилки — 0, правой — 4. Если философ проголодался, он должен сначала взять левую вилку и только затем правую. Если левая вилка уже занята другим философом, то данный философ, не беря правую вилку, дожидается момента, когда философ, взявший левую вилку закончит прием пищи и положит ее обратно на стол, после чего данный философ, взяв левую вилку, пытается взять уже правую вилку и повторяет аналогичные действия для нее, если она занята или приступает к еде, если вилка свободна. После окончания трапезы, которая длится час (секунду), философ должен вернуть на стол сначала правую вилку, а затем левую. В худшей ситуации, когда все философы решили одновременно поесть, произойдет следующее: каждый философ возьмет вилку с наименьшим номером, то есть философ №0 возьмет вилку №0, философ №1 возьмет вилку №1 и т.д., тогда вилки 0-3 будут заняты (для философа №4 левой вилкой является вилка №0, которую он ждет, так как философ №0 уже взял ее), а 4 вилка свободна. Она является правой вилкой для философа №3, который сможет свободно ее взять и начать есть. Таким образом патовая ситуация (deadlock) невозможна. Если философ не успел проголодаться, по истечению сеанса размышлений в 1.5 часа (1.5 секунды), то он продолжит размышлять не прерываясь на еду.

Философы сидят за столом до тех пор, пока в очередной сеанс размышления или приема пищи им не надоест это делать (завершение программы после  $n$  итераций;  $n$  – введенное пользователем или заданное по умолчанию число).

### 2.2 Входные данные

Входными данными является количество итераций цикла размышления или приема пищи. Вводятся через консоль и являются опциональными. Если пользователем не был введен данный аргумент, то ему по умолчанию количество итераций будет равно 3.

*Замечание:* входные данные должны представлять собой целое положительное число типа `int`. При введении некорректных данных программа выдаст соответствующее сообщение об ошибке и завершит свою работу.

### **2.3 Выходные данные**

Каждый философ выводит в консоль информацию о своем состоянии. Если он размышляет, проголодался и ждет возможности поесть или осуществляет прием пищи, соответствующая информация будет выведена в консоль. Для корректного вывода информации в консоль философ блокирует возможность вывода информации другими философами, пока данный философ не закончит вывод информации. После этого данный философ снимает ограничение на запись и продолжает свое существование.

### 3. Тестовые примеры

Корректная работа программы при одной итерации. Итерация всего одна, т.к. при большем количестве консольный вывод слишком большой для наглядного демонстрирования с помощью скриншота. (см. рис 1)

```
Philosophers will be tired after 1 cycles of thinking or eating
philosopher 0 wants to eat.
philosopher 0 started eating
philosopher 2 wants to eat.
philosopher 2 started eating
philosopher 3 wants to eat.
philosopher 4 wants to eat.
philosopher 0 finished eating
Philosopher 0 is tired and will go sleep
philosopher 4 started eating
philosopher 2 finished eating
Philosopher 2 is tired and will go sleep
philosopher 1 is thinking
Philosopher 1 is tired and will go sleep
philosopher 4 finished eating
Philosopher 4 is tired and will go sleep
philosopher 3 started eating
philosopher 3 finished eating
Philosopher 3 is tired and will go sleep
```

Рис 1. Работа программы

Обработка аргумента меньшего 1 (см. рис. 2).

```
the given argument was incorrect. Arguement should be greater than 0.
```

Рис 2. Работа программы

Обработка аргумента не являющегося целым числом типа int (см. рис. 2).

```
the given argument was incorrect. Arguement should be an integer greater than 0.
```

Рис 2. Работа программы

#### **4. Список используемых источников**

[1] Практические приемы построения многопоточных приложений. [Электронный ресурс]. // URL: <http://softcraft.ru/edu/comparch/tasks/t03/> (Дата обращения: 12.12.2020, режим доступа: свободный)

[2] Dining philosophers problem. [Электронный ресурс]. // URL: [https://en.wikipedia.org/wiki/Dining\\_philosophers\\_problem](https://en.wikipedia.org/wiki/Dining_philosophers_problem) (Дата обращения: 12.12.2020, режим доступа: свободный)

[3] Deadlock. [Электронный ресурс]. // URL: <https://en.wikipedia.org/wiki/Deadlock> (Дата обращения: 12.12.2020, режим доступа: свободный)