**Правительство Российской Федерации**

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

Факультет компьютерных наук

Департамент программной инженерии

**Отчет к домашнему заданию По дисциплине**

**«Архитектура вычислительных систем»**

Работу выполнил:

Студент группы БПИ-195 Гуницкий Р.Я.

**Москва 2020**

ОГЛАВЛЕНИЕ

[1. УСЛОВИЕ 2](#_Toc58434288)

[2. РЕШЕНИЕ 3](#_Toc58434289)

[3. ТЕСТИРОВАНИЕ 4](#_Toc58434290)

[3.1. Ввод некорректных данных 4](#_Toc58434291)

[3.2. Ввод корректных входных данных 4](#_Toc58434292)

[4. СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 7](#_Toc58434293)

# УСЛОВИЕ

Задача о каннибалах. Племя из n дикарей ест вместе из большого горшка, который вмещает m кусков тушеного миссионера. Когда дикарь хочет обедать, он ест из горшка один кусок, если только горшок не пуст, иначе дикарь будит повара и ждет, пока тот не наполнит горшок. Повар, сварив обед, засыпает. Создать многопоточное приложение, моделирующее обед дикарей. При решении задачи пользоваться семафорами.

# РЕШЕНИЕ

Перед началом работы программы пользователь вводит N - количество дикарей в племени и M - количество кусков тушеного миссионера, которое вмещает в себя горшок. В цикле создается массив потоков имитирующих действия дикарей и создается один поток, имитирующий работу повара. Входя в свою пещеру (занимает от 0 до 5 секунд) дикарь начинает искать себе еду в горшке (в консоль выводится сообщение «[Savage {{id}}] Looking for his portion») и если в горшке есть порция для него, то он берет свою порцию (уменьшает значение семафора через функцию **sem\_wait()**) и начинает трапезу (сопровождается сообщением в консоль «[Savage {{id}}] Eating»). Дикарь может есть от 1 до 3 секунд, после чего он заканчивает трапезу и выходит из пещеры (в консоль выводится сообщение «[Savage {{id}}] Finished eating»).

В случае если поиск еды в горшке не дал результатов и на данный момент времени дикарь единственный, кто не ест, то он идет к повару и начинает его будить, активируя условную переменную cv (действие сопровождается сообщением в консоль «[Savage {{id}}] Wakes up the cook»). Как только повар просыпается, он сразу уведомляет пользователя об этом сообщением «[Cook] I woke up» и начинает готовить еду. Через некоторое время (от 0,5 до 1,5 секунд) повар наполняет горшок полностью и говорит об этом пользователю, выводя сообщение в консоль «[Cook] Prepared {{ countOfPortions }} more portions» и используя функцию **sem\_post\_multiple()** увеличивает значение семафора на countOfPortions и идет спать, выводя сообщение «[Cook] I gonna sleep».

В случае если поиск еды не дал результатов и дикарь не единственный, кто остался на некоторое время без еды, то он просто ждет пока вопрос с едой решится, выводя при этом сообщение в консоль «[Savage {{id}}] Wait food». Как только повар наполняет горшок новой едой, дикарь либо начинает есть, либо, если ему так и не хватило еды, будит повара еще раз и снова ждет приготовления новых порций.

Как только все дикари закончили свою трапезу, в консоль выводится сообщение «{{N}} savage finished eating», чтобы убедиться в том, что все дикари поели.

# ТЕСТИРОВАНИЕ

## Ввод некорректных данных

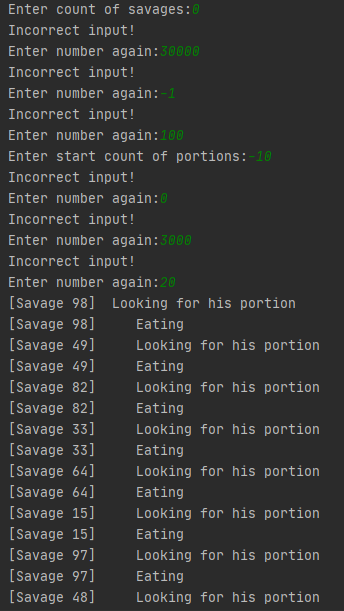


Рисунок 1. – Обработка некорректных входных данных.

## Ввод корректных входных данных

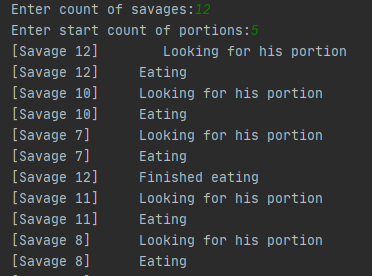


Рисунок 2. – Пришедшие дикари съедают всю еду.

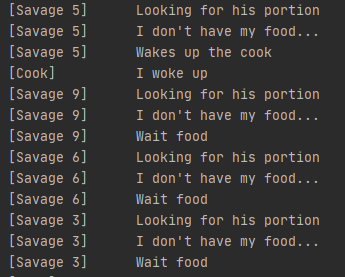


Рисунок 3. – Первый дикарь без еды будит повара, все остальные ждут.

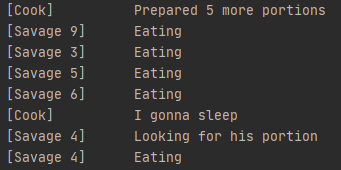


Рисунок 4. – Повар заполнил горшок едой и ушел спать, дикари едят новую еду.

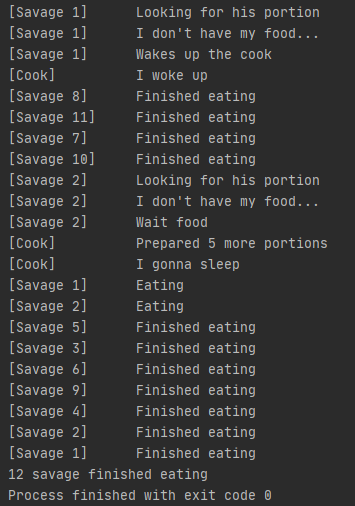


Рисунок 5. – Повтор алгоритма, вывод количества накормленных дикарей и завершение программы.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Cppreference (2020) «Документация по С++: std::condition\_variable::wait» (<https://en.cppreference.com/w/cpp/thread/condition_variable/wait>).
2. Cppreference (2020) «Документация по С++: std::mutex» (<https://ru.cppreference.com/w/cpp/thread/mutex>).
3. Docs Microsoft (2020) «Creating Threads» (<https://docs.microsoft.com/en-us/windows/win32/procthread/creating-threads>)
4. Легалов А.И.(2020) «Архитектура параллельных вычислительных систем. Многопоточность » (<http://softcraft.ru/edu/comparch/lect/07-parthread/>)
5. Легалов А.И.(2020) «Многопоточность. Простая многопоточная программа. Основные функции» (<http://softcraft.ru/edu/comparch/practice/thread/01-simple/>).
6. Легалов А.И.(2020) «Многопоточность. Синхронизация потоков. Методы синхронизации» (<http://softcraft.ru/edu/comparch/practice/thread/02-sync/>).
7. Хабр «Такие удивительные семафоры» (<https://habr.com/ru/post/261273/>)
8. Learn.info «Семафоры: введение» (<https://learnc.info/c/pthreads_semaphores.html>)