**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-194 Назмутдинов Р.Р.

Вариант 13

**Москва 2020**

**СОЖЕРЖАНИЕ**

[1. УСЛОВИЕ 3](#_Toc58460003)

[2. РЕШЕНИЕ 4](#_Toc58460004)

[3. ТЕСТИРОВАНИЕ 5](#_Toc58460005)

[3.1. Передача корректных данных 5](#_Toc58460006)

[3.2. Обработка некорректных данных 8](#_Toc58460007)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 10](#_Toc58460008)

# УСЛОВИЕ

Задача о гостинице. В гостинице 30 номеров, клиенты гостиницы снимают номер на одну ночь, если в гостинице нет свободных номеров, клиенты устраиваются на ночлег рядом с гостиницей и ждут, пока любой номер не освободится. Создать многопоточное приложение, моделирующее работу гостиницы.

# РЕШЕНИЕ

Для реализации программы использовалась стандартная библиотека C++ для работы с потоками. Вместе с этим также использовался заголовочный файл «semaphore.h» для работы с семафорами и библиотека «condition\_variable» для работы с условными переменными.

В качестве гостиницы с 30 номерами выступает семафор **sem\_t hotel**, контролирующий количество гостей в гостинице и не позволяющий заселиться в нее большему числу гостей. Для корректного вывода сообщений в консоль был создан **std::mutex messageMutex**.

Каждый гость едет в гостиницу некоторое время (от 100 миллисекунд, что меньше одного дня в программе, до 35000 миллисекунд, что равно одной неделе). Гость может приехать как на первый день, так и на седьмой. Как только гость приезжает в отель в консоль выводится сообщение «[Client {id}] Прибыл в отель». Только что прибывший гость проверяет наличие свободных мест в отеле, используя функцию **sem\_getvalue(sem\_t \*sem, int \*sval)**. В случае если мест в отеле нет, гость остается на улице и выводит в консоль следующее сообщение «[Client {id}] Придется ночевать на улице…» и ожидает освобождения комнаты. В противном же случае гость сразу заселяется в гостиницу, уменьшая значение семафора на 1 и выводя следующее сообщение в консоль «[Client {id}] Заселился в отель».

После заселения в отель каждый гость ждет начала следующего дня. Для этого используется условная переменная **std::condition\_variable newDay**. Каждый гость после заселения вызывает метод **newDay.wait(lock)** и дожидается начала нового дня. Каждый 5000 миллисекунд основной поток вызывает метод **newDay.notify\_all()**, давая всем ожидающим нового дня гостям понять, что новый день настал. По наступлению нового дня каждый гость, проживающий в отеле начинает готовиться к отъезду вместе с этим выводя сообщение «[Client {id}] Готовится к отъезду» в консоль. Подготовка к отъезду занимает не более одного дня (5000 миллисекунд). Как только поток готов выселяться из отеля он выводит сообщение «[Client {id}] Выселяется из отеля» в консоль и вызывает функцию **sem\_post(&hotel)** тем самым увеличивая значение семафора на 1 и увеличивает на единицу глобальную переменную **countThreadLeaveFromHotel**, отвечающую за количество выселившихся гостей из гостиницы.

Как только **countThreadLeaveFromHotel** становится равно количеству гостей, указанному в начале программы, основной поток выходит из цикла, объединяет все потоки в один, удаляем массив гостей и уничтожает семафор.

# ТЕСТИРОВАНИЕ

## Передача корректных данных

Для более удобного анализа работоспособности программы в гостинице будет не 30 номеров, а всего 5, что сильно уменьшит количество выводимых данных.

При запуске программы пользователя просят ввести количество ожидаемых гостей (см. рис. 1)

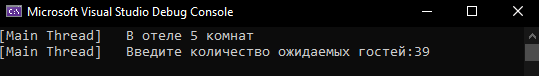


Рисунок 1. – Ввод количества гостей.

Как только пользователь вводит ожидаемое количество гостей на экран начинают выводиться итоги последующих дней (см. рис. 2 – рис. 5)

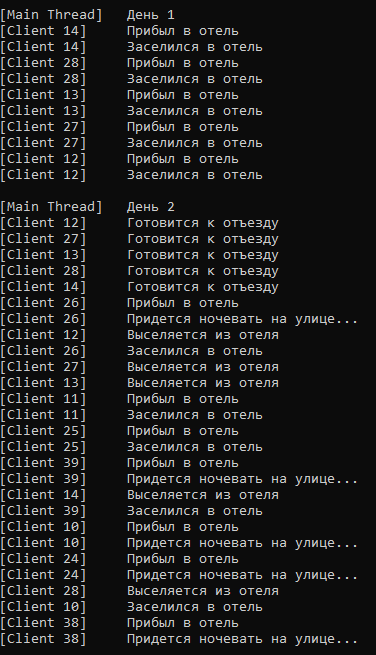


Рисунок 2. – Результаты первых двух дней.

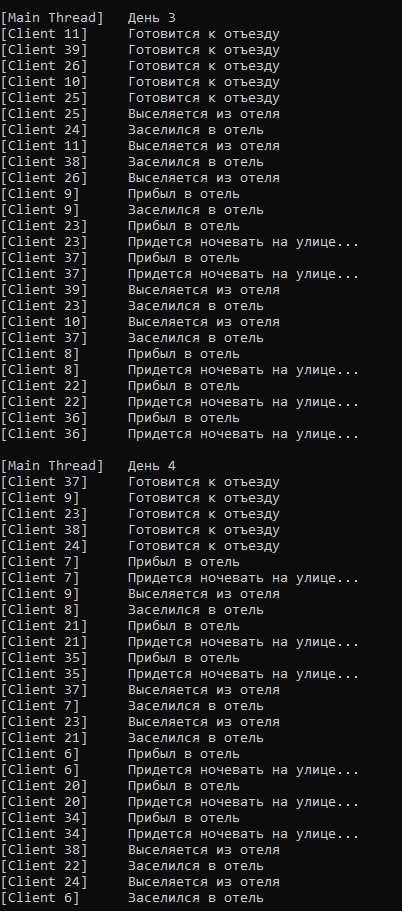


Рисунок 3. – Результаты 3-го и 4-го дней.

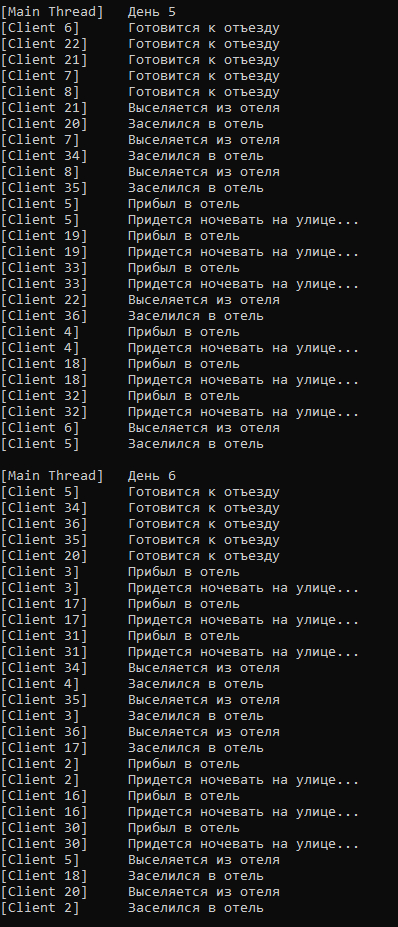


Рисунок 4. – Результат 5-го и 6-го дней.

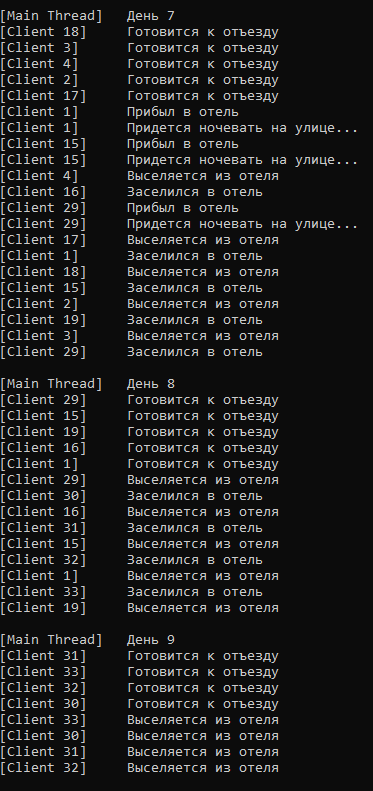


Рисунок 5. – Результаты 7-9 дней.

## Обработка некорректных данных

Программа не позволяет пользователю ввести некорректные входные данные (см. рис. 6)

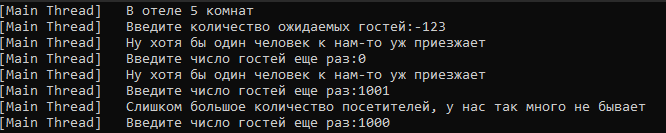


Рисунок 6. – Обработка некорректных данных

# 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>)