ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Проект по дисциплине «Архитектура вычислительных систем»

Многопоточная программа на языке C++, моделирующая работу гостиницы

Исполнитель

студент группы БПИ196-1

Махнач Ф. О.

13.12.2020 г.

Оглавление

1. Основные требования

- Оформление титульного листа: ВУЗ, департамент, название разработки, данные о студента (ФИО, номер группы/подгруппы);

- Сообщения в консоли должны отражать состояние программы и соответствовать предметной области;

- Список источников должен быть осмысленным;

- Использование библиотеки Posix thread или стандартную библиотеку потоков (thread) языка C++;

- Предусмотреть ограничение по числу итераций / времени выполнения, т. к. стандартное завершение отсутствует.

2. Текст задания

(Вариант 15)
В гостинице 10 номеров рассчитаны на одного человека и 15 номеров рассчитаны на двух человек. В гостиницу приходят клиенты дамы и клиенты джентльмены, и конечно они могут провести ночь в номере только с представителем своего пола. Если для клиента не находится подходящего номера, он уходит искать ночлег в другое место. Создать многопоточное приложение, моделирующее работу гостиницы.

3. Ремарка об условии задачи

В условии проекта указана необходимость использования семафоров и (или) условных переменных, однако при данной формулировке задания мне не удалось найти им подходящее применение. В частности, предложение «Если для клиента не находится подходящего номера, он уходит искать ночлег в другое место» не оставляет места для семафора. Если бы клиент не уходил, а продолжал ждать (хотя бы некоторое время) того, что номер освободится, семафор мог бы найти себе применение.

Помимо этого, в задаче не сказано, что джентльмены и дамы когда-либо покидают гостиницу, что делает программу достаточно прямолинейной: сначала гостиница заполняется (что происходит достаточно быстро), после чего все приходящие дамы и джентльмены не находят для себя подходящего номера.

Эти моменты достаточно сильно смутили меня, так как я не нашел указания, что можно менять условие. Я позволил себе добавить логику покидания джентльменами и дамами отеля, чтобы вывод был немного интереснее (и не противоречит условию).

4. Алгоритм решения задачи

4.1. Ввод данных

При запуске программы пользователь может задать следующие параметры:

- количество исполняемых потоков (целое число в пределах [1; макс. кол-во потоков на компьютере]);
- интервал между сообщениями в миллисекундах (целое положительное число);
- количество итераций в каждом из потоков.

4.2. Исполнение в потоке

В функции, которая выполняется каждым потоком, в цикле, ограниченном количеством итераций (см. п. 4.1.), случайно выбирается одна из операций:

1) попытаться заселить даму или джентльмена в гостиницу;
2) попытаться выселить даму или джентльмена из случайного номера.

Первая операция имеет вероятность выполнения 0,66, вторая – 0,33 (значения выбраны эмпирически).

Если был задан интервал между сообщениями (см. п. 4.1.), то после выполнения каждой итерации поток засыпает на указанное число миллисекунд.

4.3. Гостиница

Для имитации работы гостиницы было перепробовано достаточно много способов, но в конце концов я остановился на реализации, использующей счётчики:

1. Числа незанятых одноместных номеров;
2. Числа занятых дамами одноместных номеров;
3. Числа занятых джентльменами одноместных номеров;
4. Числа незанятых двухместных номеров;
5. Числа двухместных номеров с 1 жителем дамой;

6. Числа двухместных номеров с 2 жителями дамами;
7. Числа двухместных номеров с 1 жителем джентльменом;
8. Числа двухместных номеров с 2 жителями джентльменами;

Достаточно громоздко, однако другие способы (которые я пробовал) показались ещё более громоздкими. Из минусов: нельзя использовать номера комнат (комната #1, комната #10), что в принципе и не является требованием. Из плюсов: можно использовать малое число мьютексов.

Я использовал 4 мьютекса для доступа к перечисленным счётчикам:

1) для доступа к счётчикам 1-3;
2) для доступа к счётчику 4;
3) для доступа к счётчикам 5-6;
4) для доступа к счётчикам 7-8.

Группировка основана на том, какие счётчики используются вместе.
В коде это выглядит так:

```
int single_rooms_empty = SINGLE_ROOMS_NUM;
int single_rooms_with_dame = 0;
int single_rooms_with_gentleman = 0;
std::mutex single_rooms_mutex;

int double_rooms_empty = DOUBLE_ROOMS_NUM;
std::mutex double_rooms_empty_mutex;

int double_rooms_with_one_dame = 0;
int double_rooms_with_two_dames = 0;
std::mutex double_room_dames_mutex;

int double_rooms_with_one_gentleman = 0;
int double_rooms_with_two_gentlemen = 0;
std::mutex double_room_gentlemen_mutex;
```

4.4. Приезд

По приезде дамы или джентльмена происходит последовательная проверка на существование подходящей комнаты. Последовательность проверки:

- Двухместные номера с уже имеющимся жителем того же пола;
- Пустые двухместные номера;
- Пустые одноместные номера.

Функции, проверяющие возможность въезда имеют тип возвращаемого значения bool. Помимо этого, в эти функции передаётся экземпляр stringstream, хранящий сообщение, которое будет в итоге выведено на экран. Это выглядит следующим образом:

```
bool managed_to_move_in = person == Person::Dame
                          ? CheckDoubleRoomWithDame(msg)
                          : CheckDoubleRoomWithGentleman(msg);
managed_to_move_in = managed_to_move_in
                  || CheckEmptyDoubleRoom(msg, person)
                  || CheckEmptySingleRoom(msg, person);

if (!managed_to_move_in) {
    msg << ToString(person) << " haven't found a room and leaves the
hotel.\n";
}
std::cout << msg.str();
```

Сами функции проверки достаточно прямолинейны:

```
bool CheckDoubleRoomWithGentleman(std::stringstream &msg) {
    std::lock_guard<std::mutex> lock(double_room_gentlemen_mutex);
    if (double_rooms_with_one_gentleman > 0) {
        msg << "Gentleman moved into a double room with other gentleman.\n";
        --double_rooms_with_one_gentleman;
        ++double_rooms_with_two_gentlemen;
        return true;
    }
    return false;
}
```

4.5. Отъезд

Отъезд основан на случайном выборе следующих параметров:

- дама или джентльмен;
- одноместная или двухместная комната;
- если двухместная – комната с одним проживающим или с двумя.

Если оказывается, что подходящая комната не находится (например, нет одноместной комнаты, в которой живёт дама), то ничего не происходит.

5. Тестирование программы

Примеры работы программы при различных входных параметрах:

1)

```
Please, enter number of threads: -1
-1
Value 4 is invalid. Default value (4) is set.
Please, enter interval between messages (ms): -1
-1
Please, enter number of iterations per thread: -1
-1
Value 100 is invalid. Default value (100) is set.

Thread #1: Gentleman moved into an empty double room.
Thread #4: Gentleman moved into a double room with other gentleman.
Thread #3: Dame moved into an empty double room.
Thread #3: Gentleman moved into an empty double room.
Thread #3: Gentleman left double room.
Thread #3: Dame moved into an empty double room.
Thread #3: Dame moved into a double room with other dame.
Thread #3: Gentleman moved into a double room with other gentleman.
Thread #3: Dame moved into an empty double room.
Thread #4: Dame moved into a double room with other dame.
Thread #4: Gentleman moved into a double room with other gentleman.
```

2)

```
Please, enter number of threads: 2
2
Please, enter interval between messages (ms): 1
1
Please, enter number of iterations per thread: 100
100

Thread #2: Gentleman moved into an empty double room.
Thread #1: Dame moved into an empty double room.
Thread #2: Gentleman left double room. This room is now empty.
Thread #1: Dame left double room. This room is now empty.
Thread #2: Gentleman moved into an empty double room.
Thread #1: Gentleman moved into a double room with other gentleman.
Thread #2: Gentleman moved into an empty double room.
Thread #1: Dame moved into an empty double room.
Thread #2: Gentleman moved into a double room with other gentleman.
Thread #2: Gentleman left double room.
Thread #1: Dame moved into a double room with other dame.
Thread #2: Dame moved into an empty double room.
Thread #1: Gentleman moved into a double room with other gentleman.
```

3)

```
Please, enter number of threads: 4
4
Please, enter interval between messages (ms): 0
0
Please, enter number of iterations per thread: 12
12

Thread #1: Gentleman moved into an empty double room.
Thread #2: Dame moved into an empty double room.
Thread #4: Gentleman moved into a double room with other gentleman.
Thread #2: Dame moved into a double room with other dame.
Thread #1: Gentleman moved into an empty double room.
Thread #4: Gentleman moved into a double room with other gentleman.
Thread #2: Gentleman left double room.
Thread #2: Gentleman moved into a double room with other gentleman.
Thread #2: Dame moved into an empty double room.
Thread #2: Gentleman moved into a double room with other gentleman.
Thread #1: Dame moved into an empty double room.
Thread #1: Gentleman moved into an empty double room.
Thread #2: Dame left double room.
```

4)

```
Please, enter number of threads: 100
100
Value 4 is invalid. Default value (4) is set.
Please, enter interval between messages (ms): 5000
5000
Please, enter number of iterations per thread: 100
100

Thread #2: Gentleman moved into an empty double room.
Thread #2: Gentleman left double room. This room is now empty.
Thread #1: Dame moved into an empty double room.
Thread #3: Gentleman moved into an empty double room.
Thread #4: Gentleman moved into a double room with other gentleman.
Thread #1: Gentleman moved into an empty double room.
Thread #3: Dame moved into a double room with other dame.
Thread #2: Gentleman moved into a double room with other gentleman.
```

5)

```
Please, enter number of threads: 4
4
Please, enter interval between messages (ms): 100
100
Please, enter number of iterations per thread: 2
2

Thread #2: Gentleman moved into an empty double room.
Thread #1: Dame moved into an empty double room.
Thread #3: Gentleman left double room. This room is now empty.
Thread #2: Gentleman moved into an empty double room.

Process finished with exit code 0
```

6. Используемые источники

1) http://softcraft.ru/edu/comparch/practice/thread/02-sync/ – Примеры многопоточных приложений с использованием синхронизации.
2) https://en.cppreference.com/w/cpp/thread/mutex – Мьютекс стандартной библиотеки C++;
3) https://en.cppreference.com/w/cpp/thread/lock_guard – Lock guard стандартной библиотеки C++;
4) https://ravesli.com/urok-71-generatsiya-sluchajnyh-chisel-funktsii-srand-i-rand/ – генерация случайных чисел;
5) https://en.cppreference.com/w/cpp/thread/sleep_for – ожидание потоком истечения заданного времени;