

Федеральное государственное автономное образовательное учреждение  
высшего образования

**"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
"ВЫСШАЯ ШКОЛА ЭКОНОМИКИ"**

Факультет Компьютерных наук  
Департамент программной инженерии

**Пояснительная записка**

к домашнему заданию по дисциплине  
“Архитектура вычислительных систем”

Выполнил:

Студент БПИ 208

Гурдуза Даниил Михайлович

Москва 2021

## Формулировка задания:

### Вариант 10:

10. Задача о больнице. В больнице два врача принимают пациентов, выслушивают их жалобы и отправляют их или к стоматологу, или к хирургу, или к терапевту. Стоматолог, хирург и терапевт лечат пациентов. Каждый врач может принять только одного пациента за раз. Пациенты стоят в очереди к врачам и никогда их не покидают. Создать многопоточное приложение, моделирующее рабочий день клиники.

### Решение задания:

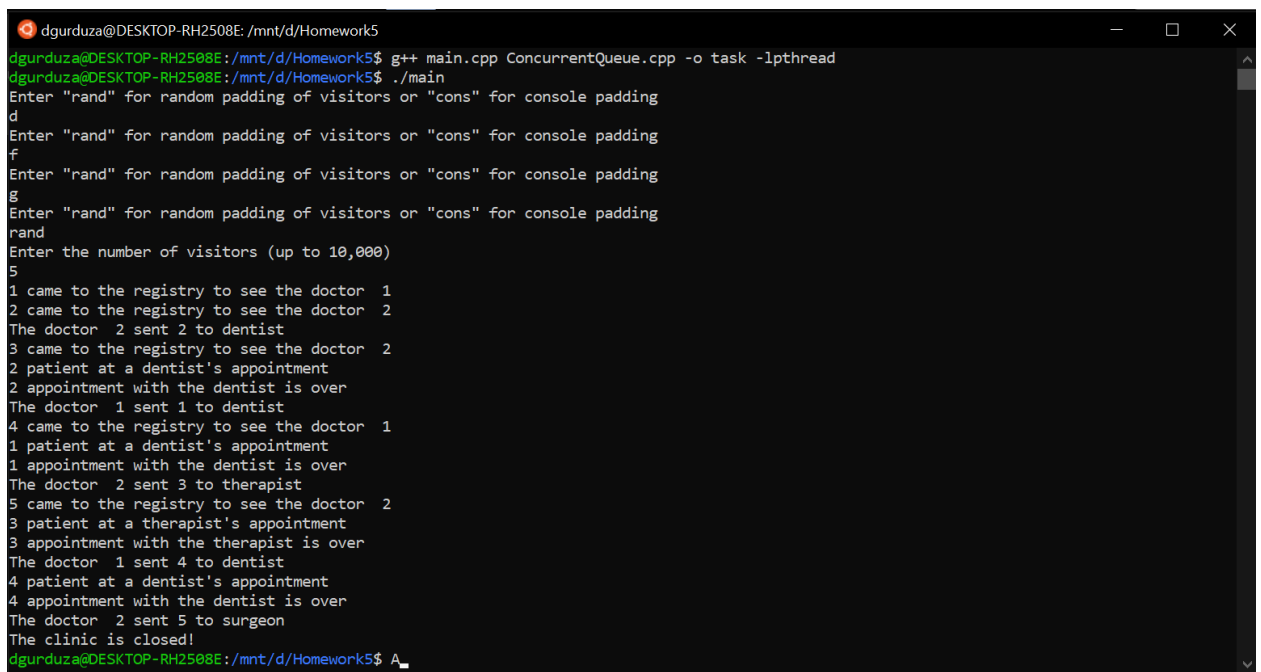
Для решения я выбрал парадигму потребители-производители. Потому что у нас есть два врача, которые заполняют очереди остальных трех врачей, те эти два врача это производители. Остальные врачи это потребители, те они считывают и изменяют(удаляют) информацию полученную от производителей. В качестве языка программирования я выбрал C++. Для многопоточности подключил библиотеку `<thread>`. Также я создал структуру данных `ConcurrentQueue` (потокобезопасная очередь). Эта структура нужна для того, чтобы обезопасить использование очередей для потребителей и производителей [\[1\]](#). Структура алгоритма такова: Пользователь вводит данные посетителей больницы либо вводит число пользователей и программа сама вводит пользователей. Далее два потока(производителя в программе это врачи в регистратуре) распределяют по трем очередям пациентов из своей общей очереди, каждая очередь относится к определенному врачу(поток). И уже параллельно выполняется работа врачей. Потоки врачей из регистратуры работают до тех пор пока есть пациенты в основной очереди. Сразу как заканчиваются пациенты в очереди в регистратуру и у каждого врача очереди пустые, потоки засыпают, а потом снова просыпаются для проверки своей очереди, и если его и основная очереди пустые поток закрывается. Для указания того, что пациент у врача я сделал засыпание потока на определенное количество времени, что имитирует работу врача.

Интересное замечание: Ожидание появления новых пациентов в 5 секунд выбрано не случайно, потому что при проверке я выяснил, что при меньшем количестве времени ожидания может вылететь ошибка при нечетных числах. У меня было 3 секунды, и программа добавляла в очередь пациента после того как поток закрылся(чаще всего это было при 5 пациентах). Из-за этого возникала ошибка при исполнении.

Начало работы: Обычный запуск через консоль и в диалоговом окне вам будет предложено ввести либо "rand" либо "cons". Первая операция обозначает ввод количества пациентов, чтобы программа сама заполнила очередь(число должно быть больше 0 и меньше 10000), вторая операция означает ручное заполнение очереди, и для того, чтобы закончить ручной ввод надо ввести "next". Если очередь пуста ввести "next" будет невозможно.

Примеры работы программы:

Ввод данных программой:



```
dgurduza@DESKTOP-RH2508E: /mnt/d/Homework5
dgurduza@DESKTOP-RH2508E: /mnt/d/Homework5$ g++ main.cpp ConcurrentQueue.cpp -o task -lpthread
dgurduza@DESKTOP-RH2508E: /mnt/d/Homework5$ ./main
Enter "rand" for random padding of visitors or "cons" for console padding
d
Enter "rand" for random padding of visitors or "cons" for console padding
f
Enter "rand" for random padding of visitors or "cons" for console padding
g
Enter "rand" for random padding of visitors or "cons" for console padding
rand
Enter the number of visitors (up to 10,000)
5
1 came to the registry to see the doctor  1
2 came to the registry to see the doctor  2
The doctor  2 sent 2 to dentist
3 came to the registry to see the doctor  2
2 patient at a dentist's appointment
2 appointment with the dentist is over
The doctor  1 sent 1 to dentist
4 came to the registry to see the doctor  1
1 patient at a dentist's appointment
1 appointment with the dentist is over
The doctor  2 sent 3 to therapist
5 came to the registry to see the doctor  2
3 patient at a therapist's appointment
3 appointment with the therapist is over
The doctor  1 sent 4 to dentist
4 patient at a dentist's appointment
4 appointment with the dentist is over
The doctor  2 sent 5 to surgeon
The clinic is closed!
dgurduza@DESKTOP-RH2508E: /mnt/d/Homework5$ A_
```

Сначала я проверил ввод неверных данных, далее ввел rand и число не превышающее 10000(поставил ограничение для оптимальности, можно снять). Ввел 5 и программа выполнилась.

## Ввод пользователей в диалоговом окне:

```
the clinic is closed.
dgurduza@DESKTOP-RH2508E:/mnt/d/Homework5$ ./main
Enter "rand" for random padding of visitors or "cons" for console padding
cons
Enter the names of the visitors. To finish typing, write "next"
a
Enter the names of the visitors. To finish typing, write "next"
v
Enter the names of the visitors. To finish typing, write "next"
b
Enter the names of the visitors. To finish typing, write "next"
next
a came to the registry to see the doctor 1
v came to the registry to see the doctor 2
The doctor 2 sent v to dentist
b came to the registry to see the doctor 2
v patient at a dentist's appointment
v appointment with the dentist is over
The doctor 1 sent a to dentist
a patient at a dentist's appointment
a appointment with the dentist is over
The doctor 2 sent b to therapist
b patient at a therapist's appointment
b appointment with the therapist is over
The clinic is closed!
dgurduza@DESKTOP-RH2508E:/mnt/d/Homework5$
```

Ввел пользователей, далее, чтобы закончить ввод надо написать next. После этой команды приложение выполнилось.

## Источники информации:

1. Цикл статей: <https://nuancesprog.ru/p/5452/>  
<https://nuancesprog.ru/p/5729/>  
<https://medium.com/nuances-of-programming/c-%D1%87%D0%B0%D1%81%D1%82%D1%8C-3-%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-%D0%BF%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%B2-%D0%B2-%D1%80%D0%B5%D1%81%D1%82%D0%BE%D1%80%D0%B0%D0%BD%D0%B5-6136f46896ee>

Это статьи про многопоточность в целом. Использовал информацию про ConcurrentQueue, condition\_variable и про std::unique\_lock<std::mutex>, и в целом помог для понимания многопоточности.

2. Про парадигму потребителей и производителей  
<https://pro-prof.com/forums/topic/parallel-programming-paradigms>
3. Про condition\_variable и многопоточность  
<https://habr.com/ru/post/182626/>