ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
“ВЫСШАЯ ШКОЛА ЭКОНОМИКИ”

Факультет компьютерных наук

Образовательная программа бакалавриата “Программная инженерия”

Микропроект 2 по дисциплине “Архитектура вычислительных систем”

Пояснительная записка

Выполнил

Студент группы БПИ191

Дадугин Егор Артемович

Москва 2020

Оглавление

1. Введение

1.1 Текст задания

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 пациента. 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.

1.2 Алгоритм программы

Работа выполнена с помощью библиотеки thread.

При написании программы была реализована потоко-безопасная очередь и функция для генерации случайного числа.

Программа реализована с помощью следующей логики:

1) Описана функция для докторов, которые осуществляют первичный осмотр пациентов. Для этих докторов существует очередь пациентов. Доктор извлекает первый элемент очереди и убирает его, сообщает о начале приема и затем назначает пациента к определенному врачу (назначение происходит случайным образом), о чем тоже сообщает выводом в консоль. Далее происходит добавление пациента в соответствующую очередь и оповещение соответствующего врача.

2) Описаны три функции для каждого врачи (стоматолог, терапевт и хирург).
В этих функциях врачи проверяют не пуста ли очередь пациентов к данному врачу. Если очередь пуста, то врачи "засыпают". Если очередь не пуста, то врачи принимают пациентов и сообщают о начале и конце приема выводом в консоль. Если общая очередь и очередь к данному врачу пуста, то врачи прекращают работу. Если "спячка врачей" длится слишком долго, то они также прекращают работу.

2. Входные и выходные данные

2.1 Входные данные

Входных данных не предусмотрено. Для ограничения работы программы было создано 9 пациентов.

2.2 Выходные данные

Программа выводит информацию о начале и конце приема каждого доктора и врача.

3. Тестирование программы

3.1 Тест 1

```
Доктор Василий принимает пациента Валерий
Доктор Борис принимает пациента Петр
Доктор Василий направил пациента Валерий к терапевту
Доктор Борис направил пациента Петр к терапевту
Доктор Борис принимает пациента Владимир
Терапевт Григорий принимает пациента Валерий
Терапевт Григорий закончил прием пациента Валерий
Терапевт Григорий принимает пациента Петр
Терапевт Григорий закончил прием пациента Петр
Доктор Борис направил пациента Владимир к хирургу
Доктор Борис принимает пациента Полина
Хирург Константин принимает пациента Владимир
Хирург Константин закончил прием пациента Владимир
Доктор Василий принимает пациента Мария
Доктор Борис направил пациента Полина к стоматологу
Доктор Борис принимает пациента Геннадий
Стоматолог Виталий принимает пациента Полина
Стоматолог Виталий закончил прием пациента Полина
Доктор Василий направил пациента Мария к стоматологу
Доктор Василий принимает пациента Людмила
Доктор Борис направил пациента Геннадий к терапевту
Доктор Борис принимает пациента Валерия
Стоматолог Виталий принимает пациента Мария
Стоматолог Виталий закончил прием пациента Мария
Терапевт Григорий принимает пациента Геннадий
Терапевт Григорий закончил прием пациента Геннадий
Доктор Василий направил пациента Людмила к терапевту
Доктор Василий принимает пациента Павел
Доктор Борис направил пациента Валерия к терапевту
Терапевт Григорий принимает пациента Людмила
Терапевт Григорий закончил прием пациента Людмила
Терапевт Григорий принимает пациента Валерия
Терапевт Григорий закончил прием пациента Валерия
Доктор Василий направил пациента Павел к стоматологу
Стоматолог Виталий принимает пациента Павел
Стоматолог Виталий закончил прием пациента Павел
Рабочий день закончен
```

Программа выводит сообщения о начале и конце приема каждого врача.

3.2 Тест 2

```
Доктор Борис принимает пациента Валерий
Доктор Василий принимает пациента Петр
Доктор Борис направил пациента Валерий к терапевту
Доктор Василий направил пациента Петр к терапевту
Доктор Борис принимает пациента Владимир
Терапевт Григорий принимает пациента Валерий
Терапевт Григорий закончил прием пациента Валерий
Терапевт Григорий принимает пациента Петр
Терапевт Григорий закончил прием пациента Петр
Доктор Борис направил пациента Владимир к стоматологу
Доктор Борис принимает пациента Полина
Доктор Василий принимает пациента Мария
Стоматолог Виталий принимает пациента Владимир
Стоматолог Виталий закончил прием пациента Владимир
Доктор Борис направил пациента Полина к терапевту
Доктор Борис принимает пациента Геннадий
Доктор Василий направил пациента Мария к терапевту
Доктор Василий принимает пациента Людмила
Терапевт Григорий принимает пациента Полина
Терапевт Григорий закончил прием пациента Полина
Терапевт Григорий принимает пациента Мария
Терапевт Григорий закончил прием пациента Мария
Доктор Василий направил пациента Людмила к терапевту
Доктор Василий принимает пациента Валерия
Доктор Борис направил пациента Геннадий к хирургу
Терапевт Григорий принимает пациента Людмила
Терапевт Григорий закончил прием пациента Людмила
Доктор Борис принимает пациента Павел
Хирург Константин принимает пациента Геннадий
Хирург Константин закончил прием пациента Геннадий
Доктор Василий направил пациента Валерия к хирургу
Доктор Борис направил пациента Павел к стоматологу
Стоматолог Виталий принимает пациента Павел
Стоматолог Виталий закончил прием пациента Павел
Хирург Константин принимает пациента Валерия
Хирург Константин закончил прием пациента Валерия
Рабочий день закончен
```

Программа выводит сообщения о начале и конце приема каждого врача.

3.3 Тест 3

```
Доктор Борис принимает пациента Валерий
Доктор Василий принимает пациента Петр
Доктор Борис направил пациента Валерий к хирургу
Доктор Василий направил пациента Петр к хирургу
Доктор Борис принимает пациента Владимир
Доктор Василий принимает пациента Полина
Хирург Константин принимает пациента Валерий
Хирург Константин закончил прием пациента Валерий
Хирург Константин принимает пациента Петр
Хирург Константин закончил прием пациента Петр
Доктор Василий направил пациента Полина к хирургу
Доктор Василий принимает пациента Мария
Доктор Борис направил пациента Владимир к терапевту
Доктор Борис принимает пациента Геннадий
Терапевт Григорий принимает пациента Владимир
Терапевт Григорий закончил прием пациента Владимир
Хирург Константин принимает пациента Полина
Хирург Константин закончил прием пациента Полина
Доктор Василий направил пациента Мария к стоматологу
Доктор Василий принимает пациента Людмила
Доктор Борис направил пациента Геннадий к хирургу
Стоматолог Виталий принимает пациента Мария
Стоматолог Виталий закончил прием пациента Мария
Доктор Борис принимает пациента Валерия
Хирург Константин принимает пациента Геннадий
Хирург Константин закончил прием пациента Геннадий
Доктор Василий направил пациента Людмила к хирургу
Доктор Василий принимает пациента Павел
Хирург Константин принимает пациента Людмила
Хирург Константин закончил прием пациента Людмила
Доктор Борис направил пациента Валерия к терапевту
Доктор Василий направил пациента Павел к хирургу
Терапевт Григорий принимает пациента Валерия
Терапевт Григорий закончил прием пациента Валерия
Хирург Константин принимает пациента Павел
Хирург Константин закончил прием пациента Павел
Рабочий день закончен
```

Программа выводит сообщения о начале и конце приема каждого врача.

4. Список использованных источников

1) Just Software Solutions [Электронный ресурс] URL: https://www.justsoftwaresolutions.co.uk/threading/implementing-a-thread-safe-queue-using-condition-variables.html (Режим доступа: свободный)
2) cppreference.com [Электронный ресурс] URL: https://en.cppreference.com/w/cpp/thread/condition_variable/wait_for (Режим доступа: свободный)

ПРИЛОЖЕНИЕ 1

ТЕКСТ ПРОГРАММЫ

```
//Дадугин Егор Артемович
//БПИ191
//Вариант 12
//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
пациента.
// 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.

#include <iostream>
#include <thread>
#include <queue>
#include <mutex>
#include <random>

//Потокобесопасная очередь
template<typename T>
class concurrent_queue {
private:
    std::queue<T> queue;
    std::mutex mutex;
public:
    void push(const T data) {
        std::unique_lock<std::mutex> locker(mutex);
        queue.push(data);
        locker.unlock();
    }

    bool empty() {
        std::unique_lock<std::mutex> locker(mutex);
        bool result = queue.empty();
        locker.unlock();
        return result;
    }

    T front() {
        std::unique_lock<std::mutex> locker(mutex);
        T result = queue.front();
        locker.unlock();
        return result;
    }

    void pop() {
        std::unique_lock<std::mutex> locker(mutex);
        queue.pop();
        locker.unlock();
    }
};

std::mutex mutex;
std::mutex mutexDentist;
std::mutex mutexTherapist;
std::mutex mutexSurgeon;
concurrent_queue<std::string> patients;
concurrent_queue<std::string> dentistPatients;
concurrent_queue<std::string> therapistPatients;
concurrent_queue<std::string> surgeonPatients;
```

```
std::condition_variable dentist;
std::condition_variable therapist;
std::condition_variable surgeon;
std::default_random_engine defEngine(time(NULL));

//Функция для генерации случайного числа
int getRand() {
    std::uniform_int_distribution<int> randomNum(0, 2);
    return randomNum(defEngine);
}

//Функция для докторов
void doctorFunc(std::string doctorName) {
    while (!patients.empty()) {
        std::unique_lock<std::mutex> locker(mutex);
        std::string patientName = patients.front();
        patients.pop();
        std::cout << "Доктор " << doctorName << " принимает пациента " <<
patientName << std::endl;
        locker.unlock();
        std::this_thread::sleep_for(std::chrono::seconds(1));
        int nextDoc = getRand();
        if (nextDoc == 0) {
            locker.lock();
            std::cout << "Доктор " << doctorName << " направил пациента " <<
patientName << " к стоматологу"
                      << std::endl;
            locker.unlock();
            //Добавляем пациента в очередь к стоматологу
            dentistPatients.push(patientName);
            //Сообщаем стоматологу о наличии пациента
            dentist.notify_all();
        } else if (nextDoc == 1) {
            locker.lock();
            std::cout << "Доктор " << doctorName << " направил пациента " <<
patientName << " к терапевту"
                      << std::endl;
            locker.unlock();
            //Добавляем пациента в очередь к терапевту
            therapistPatients.push(patientName);
            //Сообщаем терапевту о наличии пациента
            therapist.notify_all();
        } else {
            locker.lock();
            std::cout << "Доктор " << doctorName << " направил пациента " <<
patientName << " к хирургу"
                      << std::endl;
            locker.unlock();
            //Добавляем пациента в очередь к хирургу
            surgeonPatients.push(patientName);
            //Сообщаем хирургу о наличии пациента
            surgeon.notify_all();
        }
    }
}

//Функция для стоматолога
void dentistFunc(std::string doctorName) {
    while (true) {
        if (dentistPatients.empty()) {
            std::unique_lock<std::mutex> dentistLocker(mutexDentist);
            //Стоматолог ждет пациентов
            dentist.wait_for(dentistLocker, std::chrono::seconds(5));
```

```
            if (patients.empty() && dentistPatients.empty()) return;
        }

        if (!dentistPatients.empty()) {
            std::unique_lock<std::mutex> locker(mutex);
            std::cout << "Стоматолог " << doctorName << " принимает пациента
" << dentistPatients.front() << std::endl;
            std::this_thread::sleep_for(std::chrono::seconds(1));
            std::cout << "Стоматолог " << doctorName << " закончил прием
пациента " << dentistPatients.front()
                      << std::endl;
            locker.unlock();
            dentistPatients.pop();
        }
    }
}

//Функция для терапевта
void therapistFunc(std::string doctorName) {
    while (true) {
        if (therapistPatients.empty()) {
            std::unique_lock<std::mutex> therapistLocker(mutexTherapist);
            //Терапевт ждет пациентов
            therapist.wait_for(therapistLocker, std::chrono::seconds(5));
            if (patients.empty() && therapistPatients.empty()) return;
        }

        if (!therapistPatients.empty()) {
            std::unique_lock<std::mutex> locker(mutex);
            std::cout << "Терапевт " << doctorName << " принимает пациента "
<< therapistPatients.front() << std::endl;
            std::this_thread::sleep_for(std::chrono::seconds(1));
            std::cout << "Терапевт " << doctorName << " закончил прием
пациента " << therapistPatients.front()
                      << std::endl;
            locker.unlock();
            therapistPatients.pop();
        }
    }
}

//Функция для хирурга
void surgeonFunc(std::string doctorName) {
    while (true) {
        if (surgeonPatients.empty()) {
            std::unique_lock<std::mutex> surgeonLocker(mutexSurgeon);
            //Хирург ждет пациентов
            surgeon.wait_for(surgeonLocker, std::chrono::seconds(5));
            if (patients.empty() && surgeonPatients.empty()) return;
        }


        if (!surgeonPatients.empty()) {
            std::unique_lock<std::mutex> locker(mutex);
            std::cout << "Хирург " << doctorName << " принимает пациента " <<
surgeonPatients.front() << std::endl;
            std::this_thread::sleep_for(std::chrono::seconds(1));
            std::cout << "Хирург " << doctorName << " закончил прием пациента
" << surgeonPatients.front() << std::endl;
            locker.unlock();
            surgeonPatients.pop();
        }
    }
```

```
}

int main() {
    //Создание пациентов
    std::string names[9] = {"Валерий", "Петр", "Владимир", "Полина", "Мария",
"Геннадий", "Людмила", "Валерия",
                            "Павел"};
    for (int i = 0; i < 9; ++i) {
        patients.push(names[i]);
    }
    //Создание потоков для докторов
    std::thread doctor1(doctorFunc, "Василий");
    std::thread doctor2(doctorFunc, "Борис");
    std::thread dentistDoc(dentistFunc, "Виталий");
    std::thread therapistDoc(therapistFunc, "Григорий");
    std::thread surgeonDoc(surgeonFunc, "Константин");
    doctor1.join();
    doctor2.join();
    dentistDoc.join();
    therapistDoc.join();
    surgeonDoc.join();
    std::cout << "Рабочий день закончен" << std::endl;
    return 0;
}
```