Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Национальный исследовательский университет

“Высшая школа экономики”»

###### **Факультет компьютерных наук**

Программная инженерия

Исполнитель: Мурзабеков Султан

**ДОМАШНЕЕ ЗАДАНИЕ №4**

**Вариант №21**

Работа студента 2 курса бакалавриата группы БПИ-195

по предмету «Архитектура вычислительных систем»

Преподаватель:

Доктор технических наук,

Профессор

Легалов А. И

**Москва 2020**

**Задание:**

Задача про экзамен. Преподаватель проводит экзамен у группы студентов. Каждый студент заранее знает свой билет и готовит по нему ответ. Подготовив ответ, он передает его преподавателю. Преподаватель просматривает ответ и сообщает студенту оценку. Требуется создать многопоточное приложение, моделирующее действия преподавателя и студентов. При решении использовать парадигму «клиент-сервер».

**Решение.**

Ход программы написан таким образом, что на каждого студента полагается поток (если кол-во потоков >= кол-во студентов в классе). Каждый студент берет билет и садится, далее отвечает и на каждый ответ учитель выставляет оценку, после этого студент покидает класс. Создание потоков для каждого ученика реализованы с помощью библиотеки OpenMP.

Парадигма взаимодействия потоков:

Клиенты и серверы – еще один способ взаимодействия неравноправных потоков. Клиентский поток запрашивает сервер и ждет ответа. Серверный поток ожидает запроса от клиента, затем действует в соответствии с поступившим запросом.

Текст программы приведен ниже.

#include <iostream>

#include <thread>

#include <vector>

#include <mutex>

#include <sstream>

#include <string>

#include <condition\_variable>

#include <omp.h>

using namespace std;

bool finish = false; //флаг остановки экзамена

mutex serverready, exchange1, exchange2, serveranswer; //мьютексы для синхронизации

bool bserverready = false, bexchange1 = false, bexchange2 = false, bserveranswer = false; //флаги событий для синхронизации

condition\_variable cserverready, cexchange1, cexchange2, cserveranswer; //условные переменные

mutex report; //мьютекс для вывода на экран

struct

{

int stud; //номер студента

union

{

int ticket; //номер билета

int mark; //оценка

}x;

}shared; //область для обмена данными между клиентом и сервером

//вывод сообщения

void showmessage(stringstream& msg)

{

report.lock();

cout << msg.str();

msg.str("");

report.unlock();

}

//сервер

void teacher()

{

std::unique\_lock<std::mutex> lexchange1(exchange1);

std::unique\_lock<std::mutex> lexchange2(exchange2);

stringstream ss;

while (!finish) //ПРодолжаем пока не закончится экзамен

{

//установить событие что сервер(преподаватель) свободен

bserverready = true;

cserverready.notify\_one();

//Ожидаем пока кто-то из ожидающих клиентов заполнит общую область памяти

while (!bexchange1) { // цикл чтобы избежать случайного пробуждения

cexchange1.wait(lexchange1);

}

bexchange1 = false; //сбросить флаг события

ss << "Teacher start exam student " << shared.stud << " with ticket " << shared.x.ticket << endl;

showmessage(ss);

//преподаватель принимает у студента рандомное время

this\_thread::sleep\_for(chrono::milliseconds(1000 + rand() % 10000));

shared.x.mark = rand() % 3 + 3; //выставляет оценку за экзамен

ss << "Teacher marked student " << shared.stud << " with " << shared.x.mark << endl;

showmessage(ss);

//Устанавливает событие, что сервер ответил (экзамен сдан)

bserveranswer = true;

cserveranswer.notify\_one();

//ожидает пока клиент не подаст сигнал, что он прочитал переданные данные

while (!bexchange2) { // цикл чтобы избежать случайного пробуждения

cexchange2.wait(lexchange2);

}

bexchange2 = false; //сбросить флаг события

this\_thread::sleep\_for(chrono::milliseconds(10)); //небольшая задержка, прподавателю нужен отдых)))

}

}

///клиент

void stud(int n)

{

#pragma omp parallel

{

int ticket;

#pragma omp for

for (int i = 0;i < n; i++) {

std::unique\_lock<std::mutex> lserverready(serverready);

std::unique\_lock<std::mutex> lserveranswer(serveranswer);

#pragma omp critical

srand(time(nullptr));

ticket = rand() % n;

stringstream ss;

ss << "Student " << i + 1 << " has ticket " << ticket << " and wait" << endl;

showmessage(ss);

//Студент готовится рандомное время

this\_thread::sleep\_for(chrono::milliseconds(1000 + rand() % (10000 - i)));

//Когда готов, ожидает готовности сервера (преподавателя)

while (!bserverready) { // цикл чтобы избежать случайного пробуждения

cserverready.wait(lserverready);

}

#pragma omp critical

bserverready = false; //сбросить флаг события

//если готов, садится отвечать

ss << "Student " << i + 1 << " sit to answer ticket " << ticket << endl;

showmessage(ss);

//говорит серверу кто это и какой у него билет

shared.stud = i + 1;

shared.x.ticket = ticket;

//подает событие, что данные готовы

bexchange1 = true;

cexchange1.notify\_one();

//ожидает когда сервер даст ответ

while (!bserveranswer) { // цикл чтобы избежать случайного пробуждения

cserveranswer.wait(lserveranswer);

}

#pragma omp critical

bserveranswer = false; //сбросить флаг события

ss << "Student " << shared.stud << " marked for " << shared.x.mark << endl;

showmessage(ss);

//Подает сигнал серверу, что область общей памяти свободна и можно принимать следующего

bexchange2 = true;

cexchange2.notify\_one();

ss << "Student " << i + 1 << " leave room" << endl;

showmessage(ss);

}

}

}

int main()

{

int n;

//Создать поток-сервер

thread t(teacher);

//teacher();

cout << "Number of students: ";

cin >> n;

if (n <= 0) {

cout << "incorrect input data";

return 1;

}

omp\_set\_num\_threads(n);

stud(n);

finish = true; //закончить экзамен

t.join();

system("pause");

return 0;

}

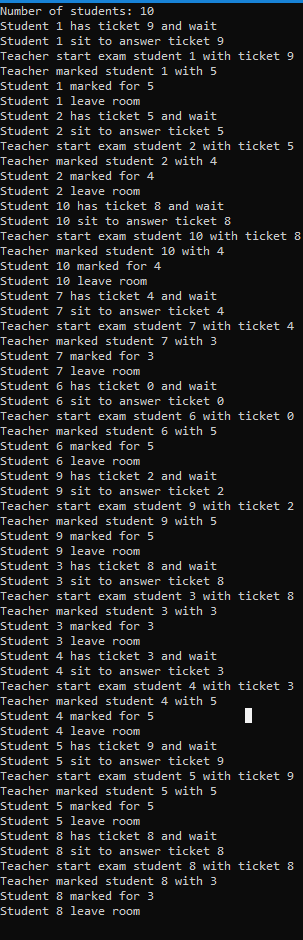


Рисунок 1 ­– Тест №1



Рисунок 2 ­– Тест №2



Рисунок 3 ­– Тест №3

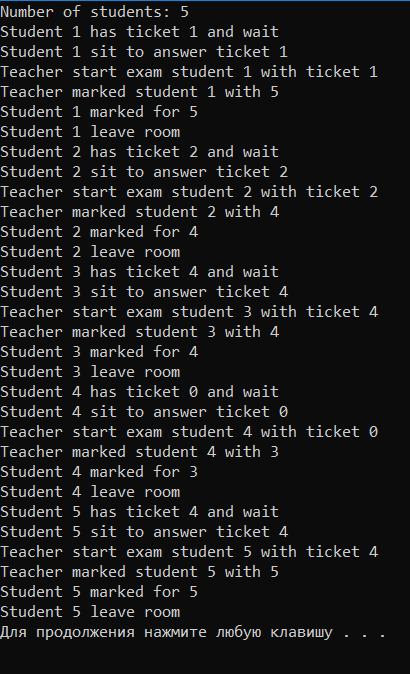


Рисунок 4 ­– Тест №4

**Список использованных источников**

1 https://nuancesprog.ru/p/5452/

2 https://stackoverrun.com/ru/q/10058625

3 https://www.osp.ru/os/2007/09/45702864

4 <https://ru.wikipedia.org/wiki/OpenMP>