ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Текст программы к микропроекту №2 по дисциплине "Архитектура вычислительных систем"

Исполнитель
студент группы БПИ195
_____ /Э. Ни/
13.12.2020 г.

```cpp
#include <iostream>
#include <unistd.h>
#include <pthread.h>

pthread_mutex_t custom_m;  //Мутекс для блокировки очереди.
pthread_mutex_t cut_m;  //Мутекс для блокировки стрижки.

int customersCnt = 10;   //Количество посетителей.
int cameCustomers = 0;  //Пришедшие посетители - количество людей в очереди.
int waitOrNot = 0;

bool isExit = true; //Проводил ли парикмахер посетителя.
bool isWait = false;    //Ожидает ли кто-то из пришедших.
bool isCutted = false;  //Пострижен или нет.

/**
 * Метод передается в поток как функция парикмахера.
 * @param num Идентификатор парикмахера.
 */
void *Hairdresser(void *num);

/**
 * Метод передается в поток как функция посетителя.
 * @param num Идентификатор посетителя.
 */
void *Customer(void *num);

int main() {
    //Генерируем тип задачи: будет ли ждать или нет.
    srand(time(nullptr));
    waitOrNot = rand() % 2;
    //Инициализируем мутексы.
    pthread_mutex_init(&custom_m, nullptr);
    pthread_mutex_init(&cut_m, nullptr);

    //Парикмахер.
    pthread_t pthhairdresser;
    //Массив потоков-посетителей.
    pthread_t pthread[customersCnt];
    //Массив номеров посетителей.
    int pthint[customersCnt];

    //Выделяем поток для парикмахера.
    std::string hName = "Нечепорчук";
    pthread_create(&pthhairdresser, nullptr, Hairdresser, (void *) (&hName));
    for (int i = 0; i < customersCnt; ++i) {
        pthint[i] = i + 1;
        pthread_create(&pthread[i], nullptr, Customer, (void *) (pthint + i));
    }
    //Отдельный цикл, потому что в программе уменьшаем customerCnt.
    for (unsigned long i : pthread)
        pthread_join(i, nullptr); //Посетитель ждем своей очереди.

    return 0;
}

void *Hairdresser(void *num) {
```

```
    const char *pName = (*((std::string *) num)).c_str();
    pthread_mutex_unlock(&cut_m);    //Открываем мутекс.
    //Пока есть запланированные посетители.
    while (customersCnt > 0) {
        //Если парикмахер проводил посетителя и некого ждать.
        while (isExit && !isWait && cameCustomers <= 0) {
            printf("Парикмахер %s: Zzzz... Ожидаю посетителя..\n", pName);
            isExit = false;
            while (!isWait);
        }
        //Обслуживаем всю очередь.
        while (cameCustomers > 0) {
            isCutted = true;
            printf("Парикмахер %s: Пришел посетителя. Стригу его.\n", pName);
            sleep(2);   //Засыпаем на время стрижки.
            --cameCustomers;
            --customersCnt;
            isCutted = false;
        }
        isWait = false;
    }
    pthread_mutex_lock(&cut_m);    //Закрываем мутекс.
}

void *Customer(void *num) {
    int pNum = *((int *) num);
    //Имитируем очередь.
    pthread_mutex_lock(&custom_m);    //Закрываем мутекс.
    sleep(waitOrNot == 0 ? 0 : 4);//Засыпаем на ожидание.
    ++cameCustomers;    //Увеличиваем количество пришедших.
    printf("Посетитель №%d: Здравствуйте, можно постричься?\n", pNum);
    isCutted = true;
    isWait = true;
    pthread_mutex_unlock(&custom_m);    //Открываем мутекс.
    pthread_mutex_lock(&cut_m);
    while (isCutted);   //Ожидание, пока идет стрижка.
    printf("Посетитель №%d: Очень нравится! Спасибо.\n", pNum);
    isExit = true;
    pthread_mutex_unlock(&cut_m);
}
```