**Факультет компьютерных наук.**

**Программная инженерия.**

Архитектура вычислительных систем.

Микропроект 2.

Пояснительная записка.

Работу выполнил Филатов Юрий (БПИ193)

Вариант 1.

**Условия задачи:**

\* В тихом городке есть парикмахерская.  
\* Салон парикмахерской мал, ходить там может только парикмахер и один посетитель.  
\* Парикмахер всю жизнь обслуживает посетителей.  
\* Когда в салоне никого нет, он спит в кресле.  
\* Когда посетитель приходит и видит спящего парикмахера, он будет его,  
\* садится в кресло и спит, пока парикмахер занят стрижкой.  
\* Если посетитель приходит, а парикмахер занят, то он встает в очередь и засыпает.  
\* После стрижки парикмахер сам провожает посетителя.  
\* Если есть ожидающие посетители, то парикмахер будит одного из них и ждет  
\* пока тот сядет в кресло парикмахера и начинает стрижку.  
\* Если никого нет, он снова садится в свое кресло и засыпает до прихода посетителя.  
\* Создать многопоточное приложение, моделирующее рабочий день парикмахерской.

**Решение задачи:**

Для решения задачи использовались потоки из библиотеки <pthread>. Так же использовались семафоры из <semaphore.h> и мутексы для posix thread.

//генерация очереди  
void \*hairCutting(void \*param) {  
 while (true) {  
 //критическая секция  
 pthread\_mutex\_lock(&mutexChair); //защита очереди  
 sem\_wait(&chair);  
 visitors[v] = v;  
 cout << "Visitor" + to\_string(v) + " садится в кресло.\n";  
 armChair = true; //имитируем, что посетитель сел в кресло  
 string str = armChair ? " сидит в кресле.\n" : "ЭТА СТРОКА НЕВОЗМОЖНА БЛАГОДАРЯ МУТЕКСАМ";  
 cout << "Visitor" + str;  
 cout << "Visitor" + to\_string(v) + " сделал прическу " + hairCuts[rand() % 12] + ".\n";  
 armChair = false; //имитируем, что посетитель встал с кресла  
 v++;  
 pthread\_mutex\_unlock(&mutexChair);  
 }  
}

***Функция haircutting*** имитирует попадание в очередь нового посетителя – потока, а смена значения переменной armChair – работу с данными, которые нужно синхронизировать и защищать от проблем с многопоточностью.

string str = armChair ? " сидит в кресле.\n" : "ЭТА СТРОКА НЕВОЗМОЖНА БЛАГОДАРЯ МУТЕКСАМ";

В частности в этой строке кода видно, что в зависимости от значения armChair, наше сообщение может передаваться корректно или некорректно, однако, мутекслок на &mutexChair не позволяет другому потоку работать параллельно в этой критической секции и изменить значение armChair, до того, как текущий поток завершит работу с ним.

sem\_wait(&chair);

семафор в свою очередь приостанавливает работу всех других потоков пока, его значение отрицательно (то есть с креслом может работать только один поток, так как изначально мы инициализируем его единицей в методе main).

Менять количество потоков-посетителей можно изменив значение переменной queueSize.

**Метод Main.**

Состоит из трех частей.

Первая часть инициализирует семафор и мутекс, а так же создает очередь потоков-посетителей:

int main() {  
 srand(time(NULL));  
  
 //инициализация мутекса и семафора  
 pthread\_mutex\_init(&mutexChair, nullptr);  
 sem\_init(&chair, 0, 1); //количество свободных ячеек равно 1 - всего одно кресло  
  
 //создание очереди  
 pthread\_t threadQueue[queueSize];

Вторая часть запускает потоки с функцией hairCutting:

for (int i = 0; i < queueSize; ++i) {  
 pthread\_create(&threadQueue[i], nullptr, hairCutting, (void \*) (i));  
}

Третья часть не дает нашему основному потоку закончиться до тех пор, пока вся очередь потоков не пройдет и не выполнит необходимый метод (так как условие while зависит от этого), а так же освобождает семафор, то есть дает следующему потоку-посетителю «выполнить стрижку»:

//пока в очереди стоят люди пропускаем их  
while (v < queueSize) {  
 sem\_post(&chair);  
}  
  
return 0;

**Бонусы:**

(Стрижки выдаются клиентам рандомно из списка реальных причесок)

Источники:

<https://learnc.info/c/pthreads_semaphores.html> - ознакомление с семафорами для pthread

<http://softcraft.ru/edu/comparch/practice/thread/02-sync/> - синхронизация потоков