Домашнее задание 07.10.2016 АиСД

студента 594 группы Бородина Максима

Робот

а)

```
#include <mutex>
#include <iostream>
#include <thread>
#include <condition_variable>
#include <atomic>

void moveLeft(std::atomic<bool>& isLeft, std::mutex& mutex, std::condition_variable& cv)
{
    std::unique_lock<std::mutex> lock(mutex);
    for (int i = 0; i < 10; ++i) {
        cv.wait(lock, [&] { return (isLeft.load()); }); // ждём, когда направление стане
        std::cout << "left" << std::endl;
        isLeft.store(false); // меняем направление следующего движения
        cv.notify_one(); // вызываем следующий поток
}
}

void moveRight(std::atomic<bool>& isLeft, std::mutex& mutex, std::condition_variable& cv
{
    std::unique_lock<std::mutex> lock(mutex);
    for (int i = 0;i < 10;++i) {
        cv.wait(lock, [&] { return (!isLeft.load()); });
        std::cout << "right" << std::endl;
        isLeft.store(true);
        cv.notify_one();
    }
```

```
};

int main()
{
    std::atomic<bool> isLeft; // флаг направления движения
    isLeft.store(true); // вначале влево
    std::mutex mutex;
    std::condition_variable cv;
std::thread t1(moveLeft, std::ref(isLeft), std::ref(mutex), std::ref(cv));
std::thread t2(moveRight, std::ref(isLeft), std::ref(mutex), std::ref(cv));
t1.join();
t2.join();
return 0;
}
```

Создаём булевский флажок, обозначающий направление следующего движения робота. Внутри каждой функции с помощью условной переменной ждём, когда направление движение станет таким, которое нам нужно, выполняем движение, меняем направление и оповещаем следующий поток, ждущий на условной переменной. Понятно, что одно движение не произойдёт два раза подряд, так как значение переменной $isLeft$ меняется на нужное нам только после противоположного движения.

б)

```
#include <mutex>
#include <iostream>
#include <thread>
#include <condition_variable>
#include "semaphore.h"


void moveLeft(Semaphore& sem1, Semaphore& sem2)
```

```
{
for (int i = 0; i < 10; ++i) {
sem2.wait();
        std::cout << "left" << std::endl;
sem1.post();
}
}

void moveRight(Semaphore& sem2, Semaphore& sem1)
{
    for (int i = 0;i < 10;++i) {
        sem1.wait();
        std::cout << "right" << std::endl;
        sem2.post();
    }
};

int main()
{
    Semaphore semaphore1;
    Semaphore semaphore2;
semaphore2.post();
std::thread t1(moveLeft, std::ref(semaphore1), std::ref(semaphore2));
std::thread t2(moveRight, std::ref(semaphore2), std::ref(semaphore1));
t1.join();
t2.join();
return 0;
}
```

Итак, мы создаём два "скрещенных"семафора. Каждый поток при выполнении вызывает *wait*() одного семафора, а по завершении, *post*() у другого. Изначально

в первом семафоре значение счётчика = 1, в другом 0. После каждого выполнения счётчики меняются между собой значениями. Таким образом, потоки не смогут одновременно выполняться. Так же, один поток не сможет выполниться два раза, так как для этого нужно, чтобы значение переменной у какого-то из семафоров было хотя бы 2, чего произойти не может