Задача “Шагающий робот", Ткаченко Дмитрий, группа 595.

Пункт (а)

Сначала кратко опишем суть кода: с помощью флага будем следить за направлением следующего движения. Будем ожидать нужного нам направления движения с помощью condition variable, при нужном значении выполняем шаг, меняем condition variable и сигналим в другой поток. В таком случае одно движение не может произойти два раза подряд, так как значение condition variable меняется на требуемое только после другого движения.

```
void stepLeft(std::atomic<bool> &dir, std::mutex &mutex, std::condition_variable &cv) {
    std::unique_lock<std::mutex> lock(mutex);
    for (int i = 0; i < steps_count; i++) {
        cv.wait(lock, [&] {return (dir.load()); }); // waiting for our direction
        //Some function that required in left step.
        dir.store(false); //change the direction
        cv.notify_one(); //signal to next thread
    }
}
```

Согласно словесному описанию и примеру процедуры stepLeft легко написать процедуру stepRight.

```
int main(int argc, const char * argv[]) {

    std::mutex mutex;
    std::condition_variable cv;
    std::atomic<bool> dir;
    dir.store(true); //true - left direction, false - right direction.

    std::thread th1(stepLeft, std::ref(dir), std::ref(mutex), std::ref(cv));
    std::thread th2(stepRight, std::ref(dir), std::ref(mutex), std::ref(cv));
    th1.join(); th2.join();
    return 0;
}
```

Пункт (б)

Нам понадобится два семафора. При выполнении каждый поток будет делать wait() на одном из семафоров, после этого делая post() на другом. Начальные значения семафоров суть 1 и 0 соответственно. Как мы обеспечим раздельное выполнение? После каждой итерации будем менять значения счетчиков местами. В силу того, что оба счетчика меньше 2, то никакой поток не выполнится подряд два или более раз.

```
int main(int argc, const char * argv[]) {
    semaphore sem1, sem2(1);
    std::thread th1(stepLeft, std::ref(sem1), std::ref(sem2));
    std::thread th2(stepRight, std::ref(sem1), std::ref(sem2));
    th1.join(), th2.join();
    return 0;
}
```

```
void stepLeft(semaphore &sem1, semaphore &sem2) {
    for (int i = 0; i < steps_count; i++) {
        sem2.wait();
        //Some function that required in left step.
        sem1.post();
    }
}

void stepRight(semaphore &sem1, semaphore &sem2) {
    for (int i = 0; i < steps_count; i++) {
        sem1.wait();
        //Some function that required in right step.
        sem2.post();
    }
}
```