6.13 编写一个有限缓冲的管程，缓冲区嵌在管程中：

class Monitor {

private:

std::mutex mtx;

std::condition\_variable cv\_1, cv\_2;

constexpr static int initSize = 10;

int arr[initSize];

int count = 0;

int in = 0, out = 0;

public:

void push(const int cnt) {

std::unique\_lock<std::mutex> lck(mtx);

if(count == initSize)

cv\_1.wait(lck);

std::cout << "push" << '\n';

++count;

arr[in] = cnt;

in = (in + 1) % initSize;

if(count not\_eq 0)

cv\_2.notify\_all();

}

int pop() {

std::unique\_lock<std::mutex> lck(mtx);

if(count == 0)

cv\_2.wait(lck);

std::cout << "pop" << '\n';

--count;

int res = arr[out];

out = (out + 1) % initSize;

if(count not\_eq initSize)

cv\_1.notify\_all();

return res;

}

} monitor;

void producer() {

static int cnt = 0;

monitor.push(++cnt);

}

std::map<int, int> times;

void consumer() {

auto cnt = monitor.pop();

std::cout << cnt << "\n";

++times[cnt];

}

int main() {

std::thread produce[30];

std::thread consume[30];

for(int i = 0;i < 30; ++i) {

produce[i] = std::thread(producer);

consume[i] = std::thread(consumer);

}

for(int i = 0;i < 30; ++i) {

produce[i].join();

consume[i].join();

}

for(const auto it : times)

std::cout << it.first << "\t" << it.second << " 次\n";

return 0;

}