```cpp
    template<typename T>
    class message_queue
    {
       std::mutex _m;
       std::queue<T> _q;

        using lock_guard = std::lock_guard<std::mutex>;

    public:
        using value_type = T;

        message_queue() = default;
        message_queue(const message_queue& ) = delete;
        message_queue& operator=(const message_queue&) = delete;


        void push(T item)
        {
            lock_guard lk{_m};
            _q.push(std::move(item));
        }

      bool pop(T& t)
        {
            lock_guard lk{_m};
            if(_q.empty())
                return false;

            t = std::move(_q.front());
            _q.pop();
            return true;
        }
```

```cpp
template<typename T>
class message_queue
{
   std::mutex _m;
   std::queue<T> _q;

    using lock_guard = std::lock_guard<std::mutex>;

public:
    using value_type = T;

    message_queue() = default;
    message_queue(const message_queue& ) = delete;
    message_queue& operator=(const message_queue&) = delete;


    void push(T item)
    {
        lock_guard lk{_m};
        _q.push(std::move(item));
    }

  bool pop(T& t)
    {
        lock_guard lk{_m};
        if(_q.empty())
            return false;

        t = std::move(_q.front());
        _q.pop();
        return true;
    }
```