37

```cpp
template<typename Type>
class synchronized_value
{
public:
    synchronized_value(const synchronized_value&) = delete;
    synchronized_value &operator=(const synchronized_value&) = delete;

    template<typename... Args>
    synchronized_value(Args&&... args)
        : val (std::forward<Args> (args)...)
    {}

    template<typename Fn, typename Up, typename... Types>
    friend std::invoke_result_t<Fn, Up&, Types&...> apply (Fn&&, synchronized_value<Up>&,
                                                           synchronized_value<Types>&...);

private:
    std::mutex mutex;
    Type val;
};
```

synchronized_value

```
template<typename T>
struct is_sync<synchronized_value<T>> : std::true_type
{};
```

synchronized_value

```
template<typename Type>
class synchronized_value
{
public:
    synchronized_value(const synchronized_value&) = delete;
    synchronized_value &operator=(const synchronized_value&) = delete;

    template<typename... Args>
    synchronized_value(Args&&... args)
        : val (std::forward<Args> (args)...)
    {}

    template<typename Fn, typename Up, typename... Types>
    friend std::invoke_result_t<Fn, Up&, Types&...> apply (Fn&&, synchronized_value<Up>&,
                                                           synchronized_value<Types>&...);

private:
    std::mutex mutex;
    Type val;
};
```

```
template<typename T>
struct is_sync<synchronized_value<T>> : std::true_type
{};
```