Synchronization and Ordering

C++ has six different memory orderings.

```
enum memory_order {
    memory_order_relaxed,
    memory_order_consume,
    memory_order_acquire,
    memory_order_release,
    memory_order_acq_rel,
    memory_order_seq_cst
};
```

- Sequential consistency is the default.
  - The memory model for C# and Java.
  - `memory_order_seq_cst`
  - Implicit argument for atomic operations.

```
std::atomic<int> shared;

shared.load() ≅ shared.load(std::memory_order_seq_cst);
```

Synchronization and Ordering

To systemize the memory orderings, you must answer two questions.

1. For which kind of operations should you use which memory ordering?
2. Which synchronization and ordering constraints are defined by the various memory orderings?

Synchronization and Ordering

1. For which kind of operations should you use which memory ordering?

- **read** operations:

  `memory_order_acquire` and `memory_order_consume`

- **write** operations:

  `memory_order_release`

- **read-modify-write** operations:

  `memory_order_acq_rel` and `memory_order_seq_cst`

! `memory_order_relaxed` doesn't define synchronization and ordering constraints

Synchronization and Ordering

Operation	read	write	read-modify-write
test_and set			yes
clear		yes	
is_lock_free	yes		
load	yes		
store		yes	
exchange			yes
compare_exchange_weak compare_exchange_strong			yes
fetch_add, += fetch_sub, -=			yes
++, --			yes

```
std::atomic<int> atom;
atom.load(std::memory_order_acq_rel)     atom.load(std::memory_order_acquire)
atom.load(std::memory_order_release)  ≅  atom.load(std::memory_order_relaxed)
```

Synchronization and Ordering

2. Which synchronization and ordering constraints are defined by the various orderings?

- **Sequential consistency**
  - Global ordering of all threads

    `memory_order_seq_cst`
- **Acquire-release semantics**
  - Ordering between read and write operations on the same atomic

    `memory_order_consume`, `memory_order_acquire`, `memory_order_release`, and `memory_order_acq_rel`
- **Relaxed semantics**
  - No synchronization and ordering constraints

    `memory_order_relaxed`