Lightweight Data Race Detection

C

C

h

e

C

k

S

t

a

t

e

d

a

O

C

S

n

m

r

e

a

d

S

{

O

S

W

t

n
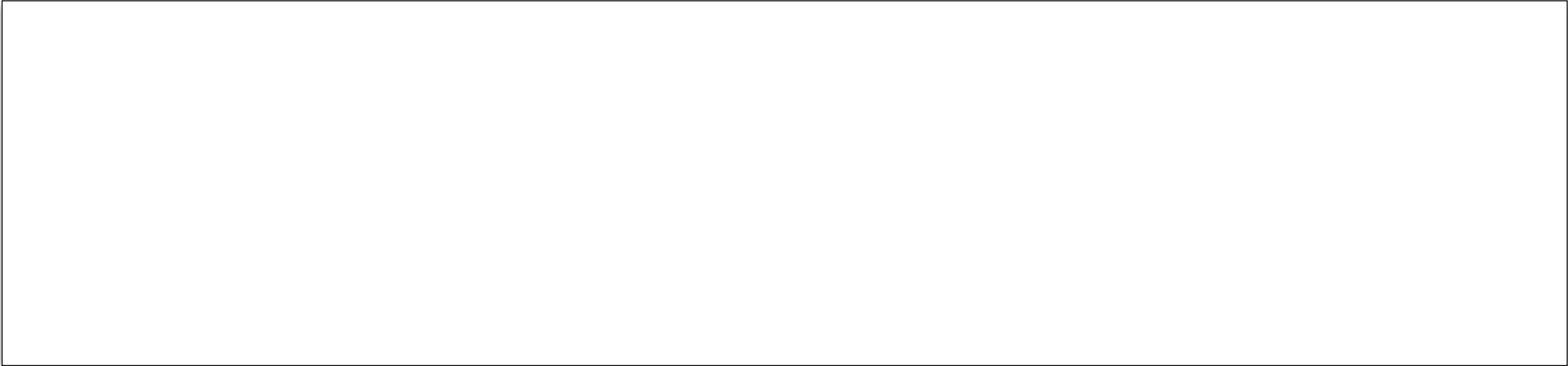
g

T

■
』

}

■
❞

```cpp
void read_started (check_state& state)
{
    ++state.num_readers; // must be first

    if (state.is_writing)
        std::terminate();
            // read during active write
}


void write_started (check_state& state)
{
    // must be first
    if (state.is_writing.exchange (true))
        std::terminate();
            // write during active write

    if (state.num_readers > 0)
        std::terminate();
            // write during active read
}
```

```cpp
void read_ended (check_state& state)
{
    --state.num_readers;
}

void write_ended (check_state& state)
{
    state.is_writing = false;
}
```

```cpp
struct check_state
{
    std::atomic<size_t> num_readers { 0 };
    std::atomic<bool> is_writing { false };
};
```

```cpp
if constexpr (type == check_type::read)
```

`template`<check_type type>

read,

```
} ;
```

`enum class` check_type

```
: state (check_state)
```

struct scoped_check

```
scoped_check (check_state& check_state)
```

{

```
write_started (state);
```

```
read_started (state);
```

}

`~scoped_check()`

```
write_ended (state);
```

```
check_state& state;
```

```
read_ended (state);
```

Lightweight Data Race Detection

```cpp
std::atomic<bool> is_writing { false };
```

`struct` check_state

```cpp
std::atomic<size_t> num_readers { 0 };
```

} ;

```cpp
++state.num_readers; // must be first
```

```cpp
void read_started (check_state& state)
```

```
if (state.is_writing)
```

```
std::terminate();
```

```
// read during active write
```

```cpp
void write_started (check_state& state)
```

// must be first

```
// write during active write
```

```
if (state.num_readers > 0)
```

```
// write during active read
```

```
if (state.is_writing.exchange (true))
```
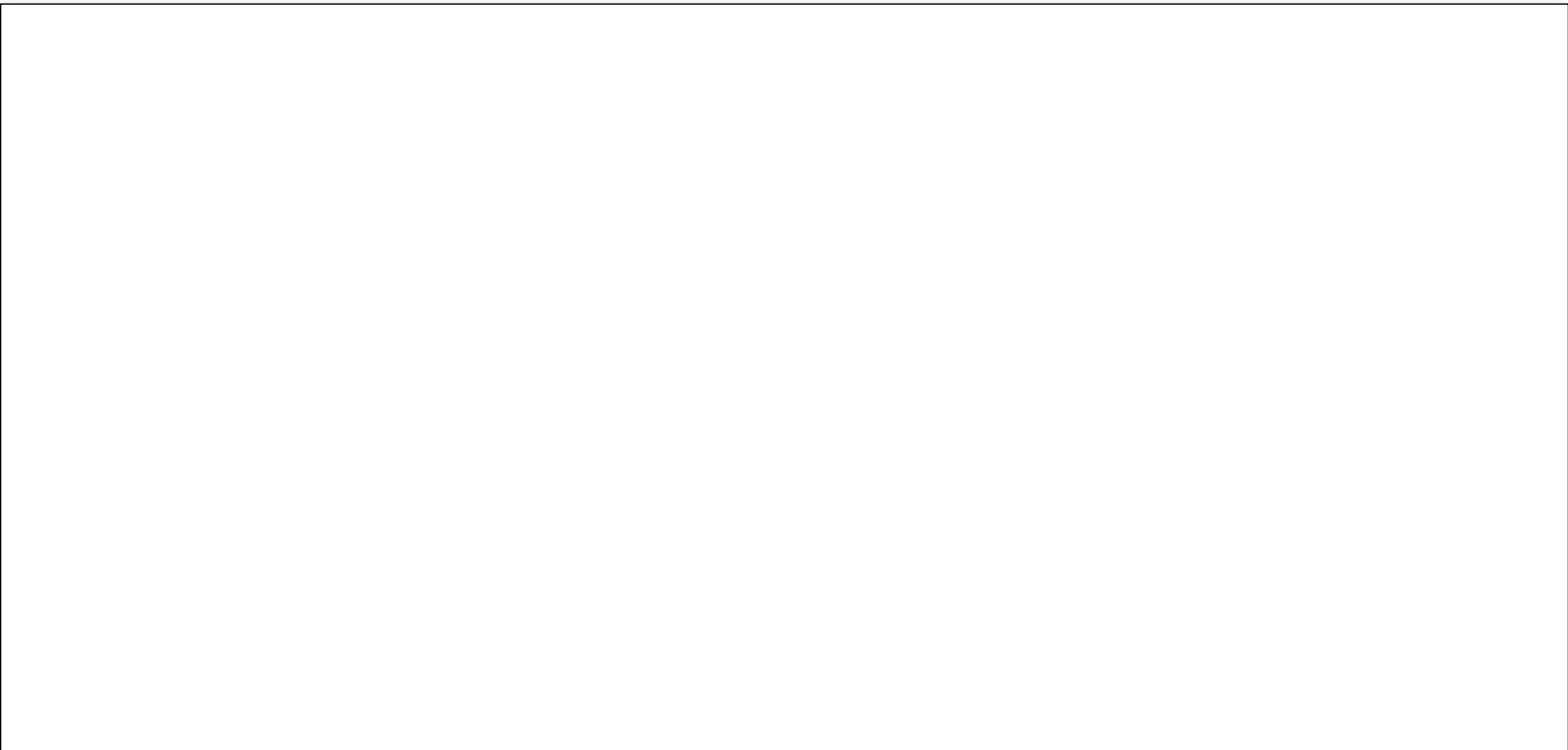
{

```cpp
void write_ended (check_state& state)
```

```cpp
void read_ended (check_state& state)
```

```
--state.num_readers;
```

```
state.is_writing = false;
```