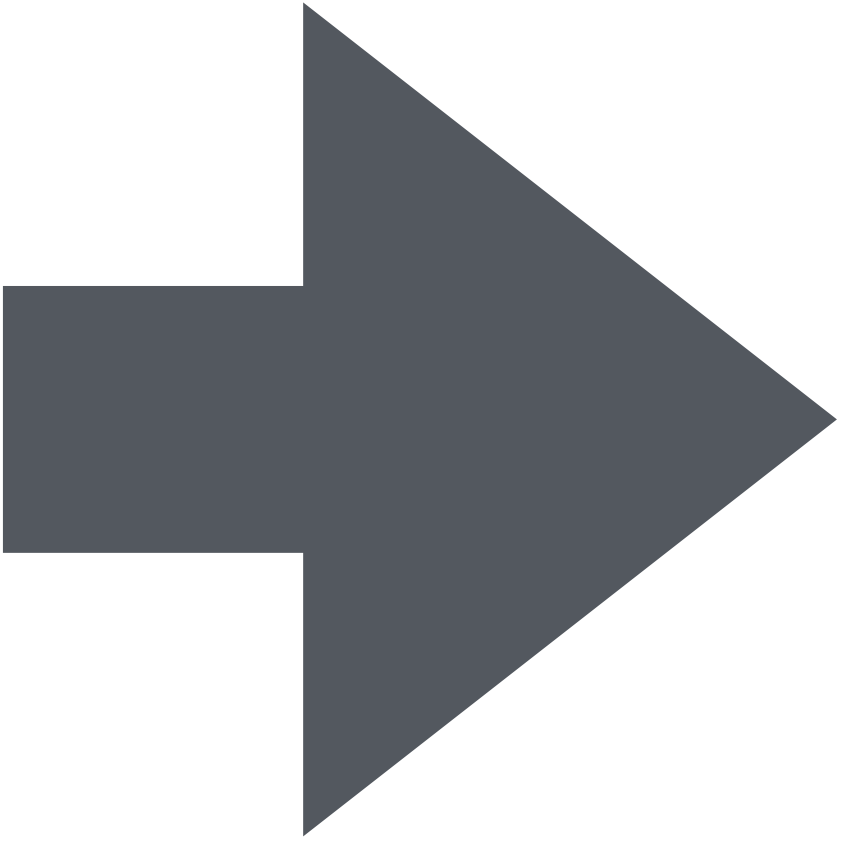
Implicit `mutex` locking

65

```
class person(mutex)
{
public:
    person() = default;

    std::string get_first_name() const
    {
        return first_name;
    }

    void set_first_name (std::string_view new_first)
    {
        first_name = new_first;
    }

    // Repeat for last_name

private:
    std::string first_name, last_name;
};
```

```
class person
{
public:
    person() = default;

    std::string get_first_name() const
    {
        std::scoped_lock _ (mutex);
        return person_.get_first_name();
    }

    void set_first_name (std::string_view new_first)
    {
        std::scoped_lock _ (mutex);
        person_.set_first_name (new_first);
    }

    // Repeat for last_name

private:
    class __person;
    std::mutex mutex;
    mutable __person person_;
};

template<>
struct is_sync<person> : std::true_type {};
```




mutex

Implicit

locking

shared_mutex

person(mutex

std::string

first_name

set_first_name

first_name,

(std::string_view

get_first_name()

```
new_first;
```

last_name

new_first)

```
last_name;
```

```
first_name;
```

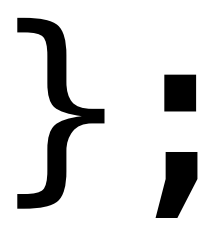
public:

private:

void

person()

const



class


default;


for

return

Repeat

[redacted]

[redacted]

person(shared_mutex

std::scoped_lock

std::mutex

person_.set_first_name

std::true_type

get_first_name()

(mutex);

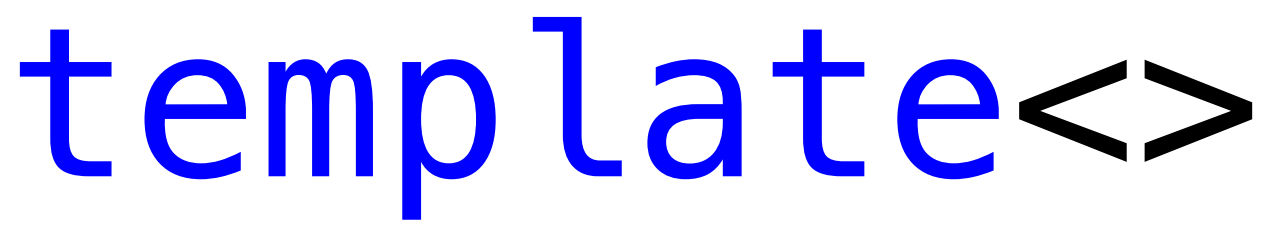
set_first_name

person()

std::string

is_sync<person>


default;


template<>

last_name

new_first)

```
person_.get_first_name();
```

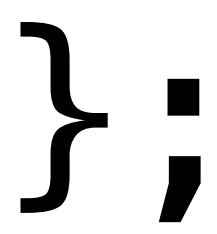
(std::string_view


__person;

```
(new_first);
```

```
person_;
```

void




mutex;

struct

private:


const

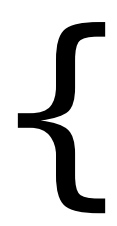
[redacted]

return

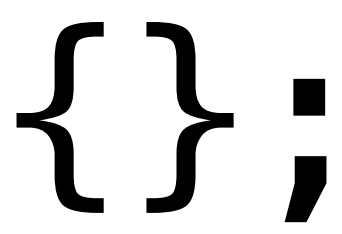
class

__person

mutable




for




public:

Repeat

person

class

std::shared_mutex

std::shared_lock

std::unique_lock

locking

66

mutex;