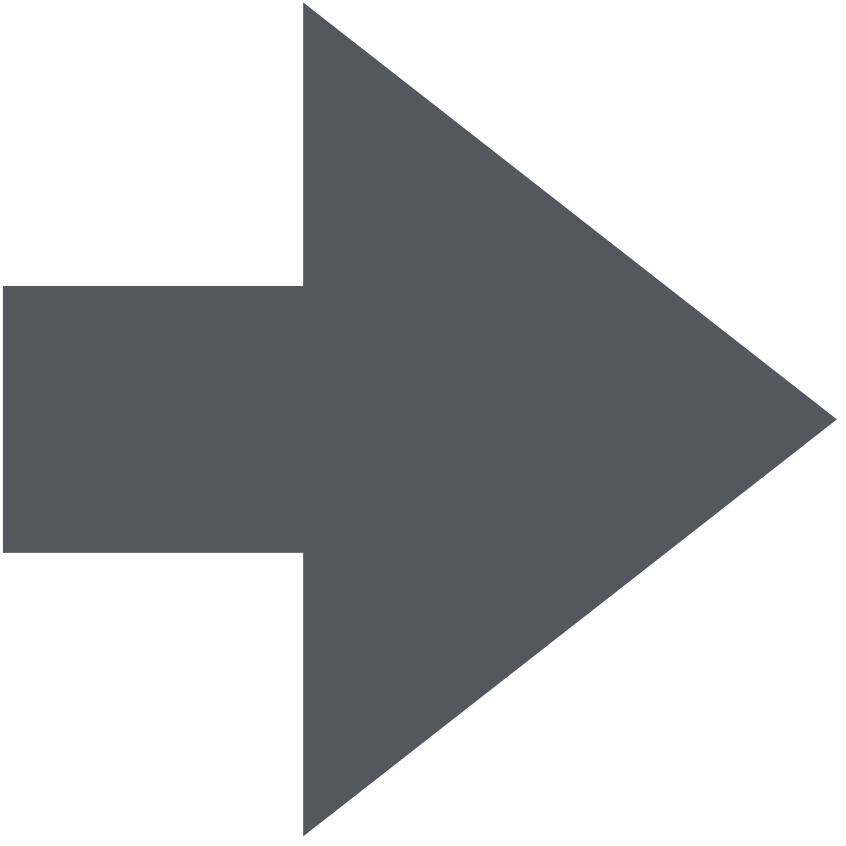
Implicit `mutex` locking

107

```
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_internal.get_first_name();
    }

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

    // Repeat for last_name

private:
    struct person_internal;
    std::mutex mutex;
    mutable person_internal person_internal;
};

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




mutex

locking

Implicit

shared_mutex

person(mutex

set_first_name

std::string

get_first_name()

```
first_name;
```

(std::string_view

```
new_first;
```

first_name,

last_name

```
last_name;
```

const

Repeat

person()

new_first)

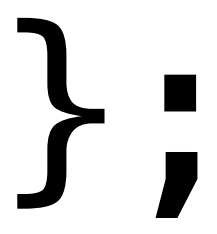

default;

return

first_name

void

private:



public:

class


for

[redacted]

[redacted]

person(shared_mutex

std::mutex

std::scoped_lock

is_sync<person>

set_first_name

```
person_internal.get_first_name();
```

get_first_name()

person_internal

person_internal.set_first_name

```
person_internal;
```

std::string

```
(new_first);
```

```
person_internal;
```

(std::string_view

std::true_type

person()

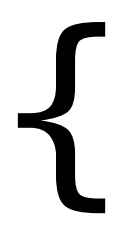
mutable

private:

(mutex);

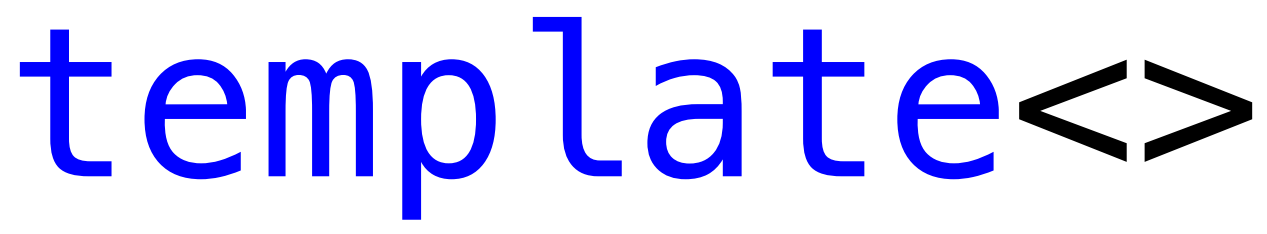
[redacted]


default;




const

return


template<>


mutex;

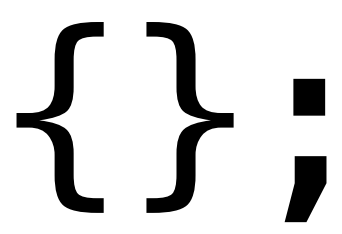
class

last_name

Repeat

new_first)


public:



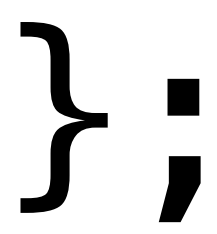

for

person

struct

void

struct



std::shared_lock

std::unique_lock

std::shared_mutex

locking

108

mutex;