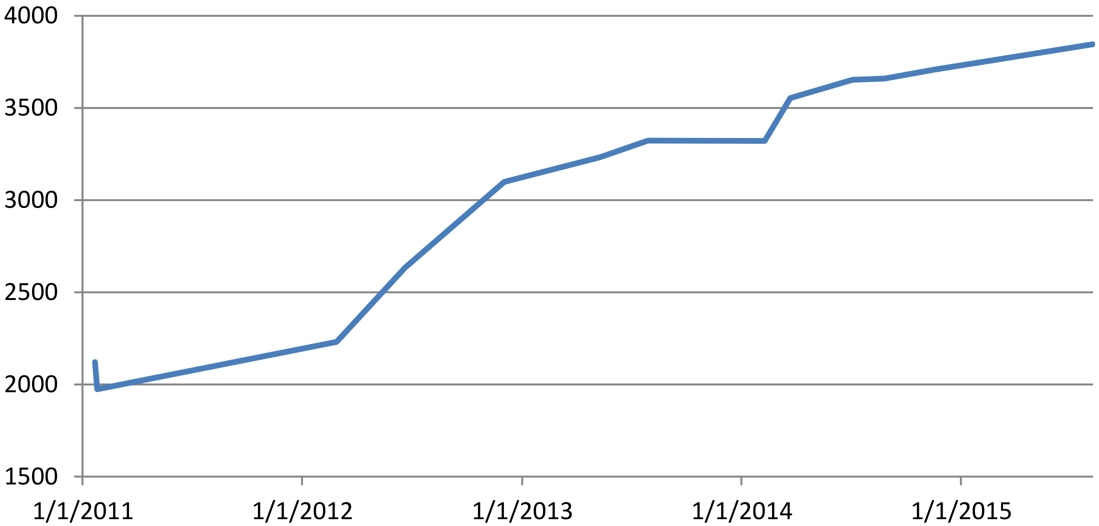
cppreference.com

the community wiki

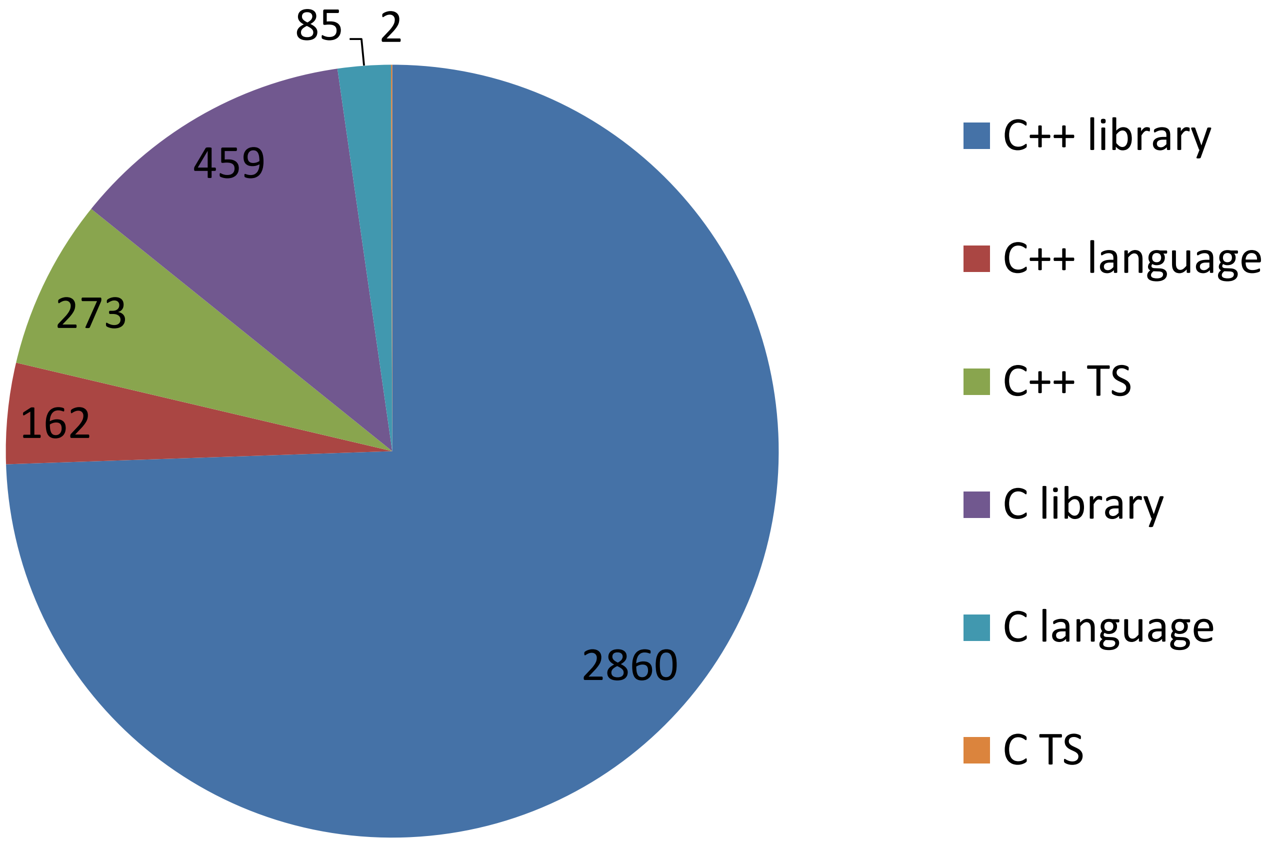
How big is it?



Content pages

Peak month	Unique visitors	Number of visits	Pages	Hits	Bandwidth
Mar-15	556613	1357221	20160329	25294838	943.32 GB

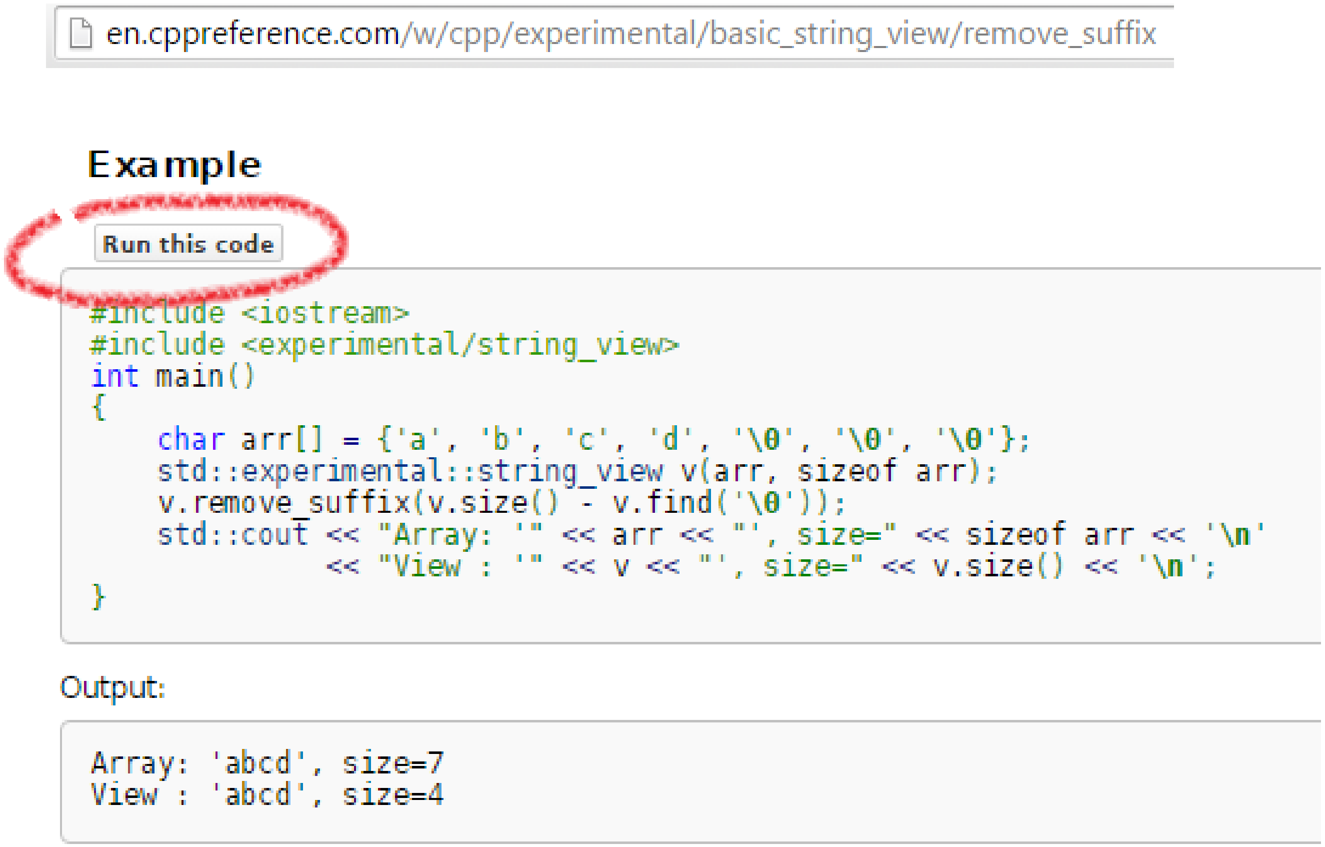
3841 pages of content



Not just text

- Open, freely editable wiki
- Offline archives
  - wget dump, HTML book
  - Qt and DevHelp books (Debian, Ubuntu, Arch)
  - Doxygen tags
  - man pages
- Inline compiler
- Revision control

Inline compiler



en.cppreference.com/w/cpp/experimental/basic_string_view/remove_suffix

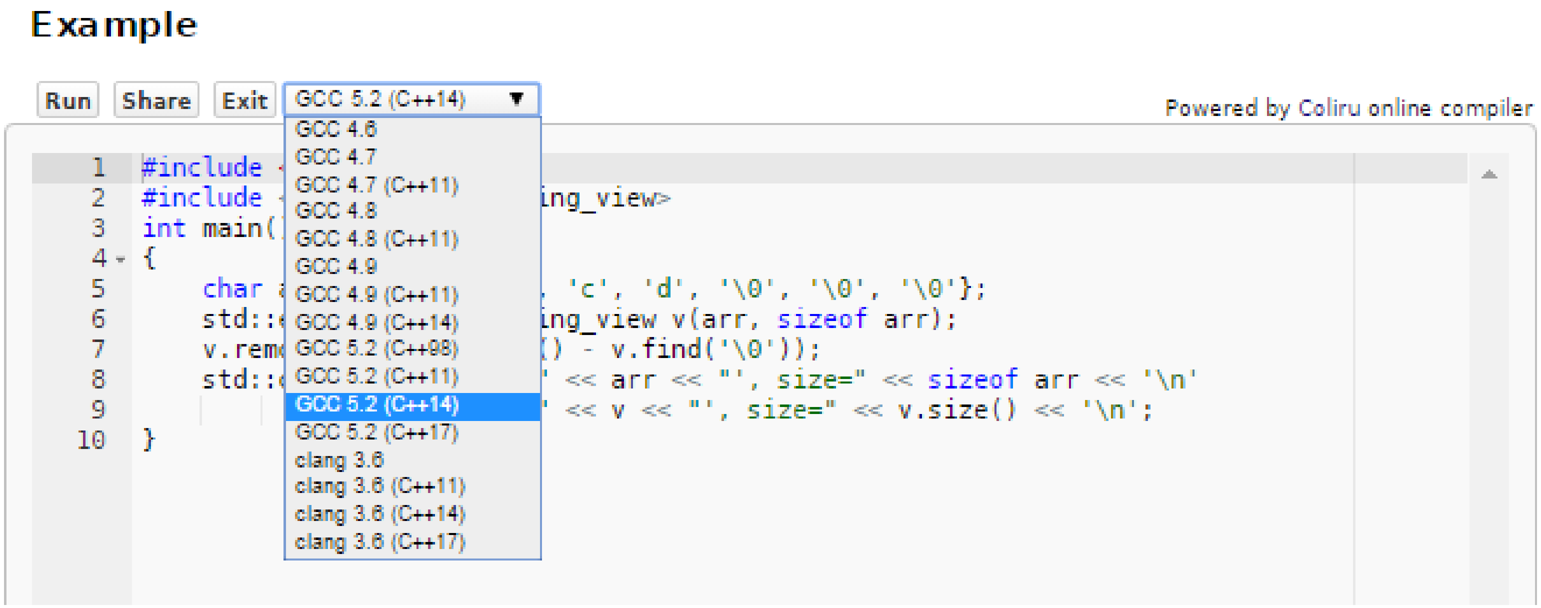
Example

```
#include <iostream>
#include <experimental/string_view>
int main()
{
    char arr[] = {'a', 'b', 'c', 'd', '\0', '\0', '\0'};
    std::experimental::string_view v(arr, sizeof arr);
    v.remove_suffix(v.size() - v.find('\0'));
    std::cout << "Array: '" << arr << "', size=" << sizeof arr << '\n'
              << "View : '" << v << "', size=" << v.size() << '\n';
}
```

Output:

```
Array: 'abcd', size=7
View : 'abcd', size=4
```

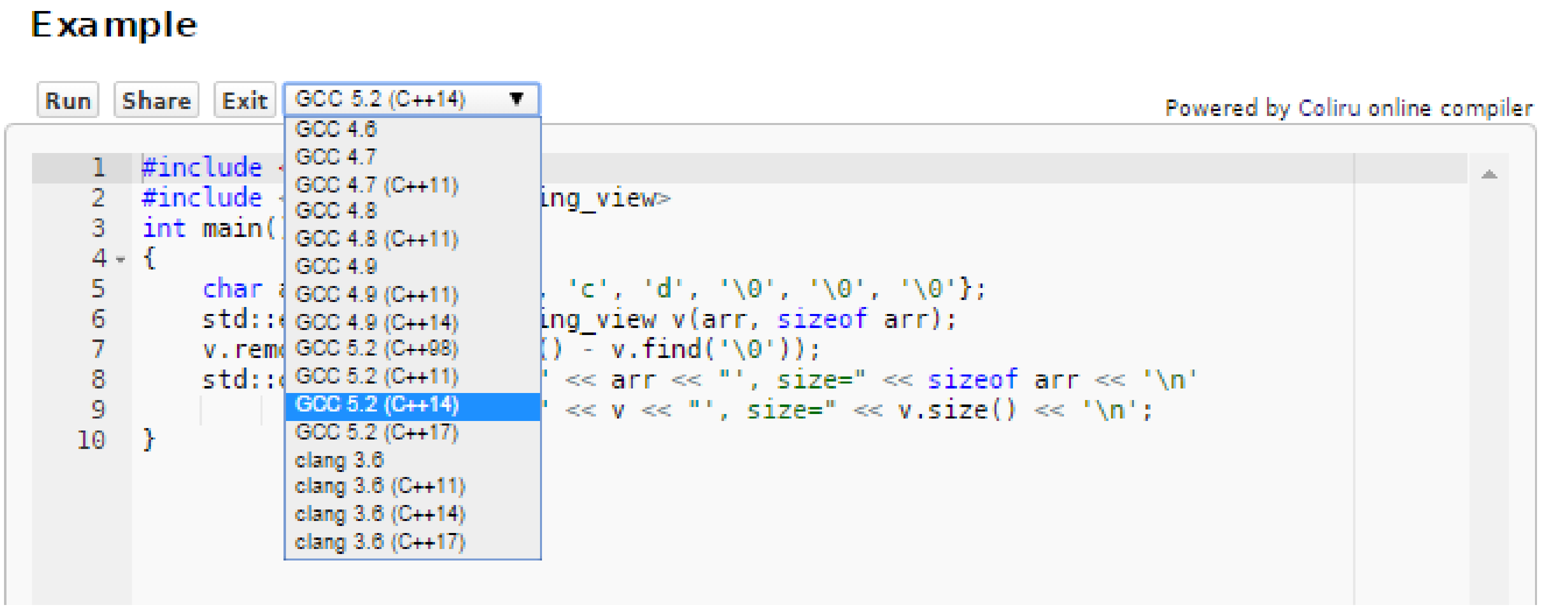
Inline compiler



Example

```
#include <
#include <                 ing_view>
int main(
{
    char a                , 'c', 'd', '\0', '\0', '\0'};
    std::e                ing_view v(arr, sizeof arr);
    v.remo                () - v.find('\0'));
    std::c                ' << arr << "', size=" << sizeof arr << '\n'
                          ' << v << "', size=" << v.size() << '\n';
}
```

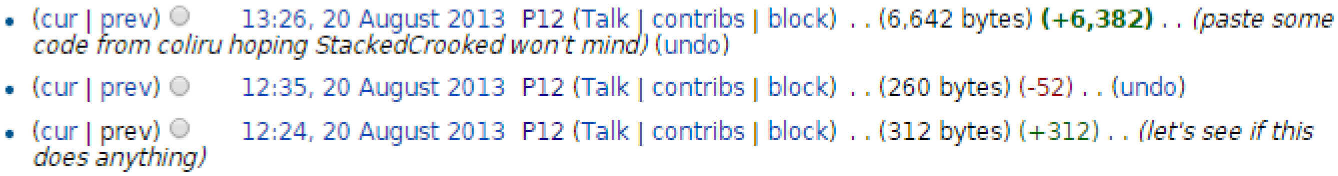
Inline compiler



GCC 5.2 (C++14)

Powered by Coliru online compiler

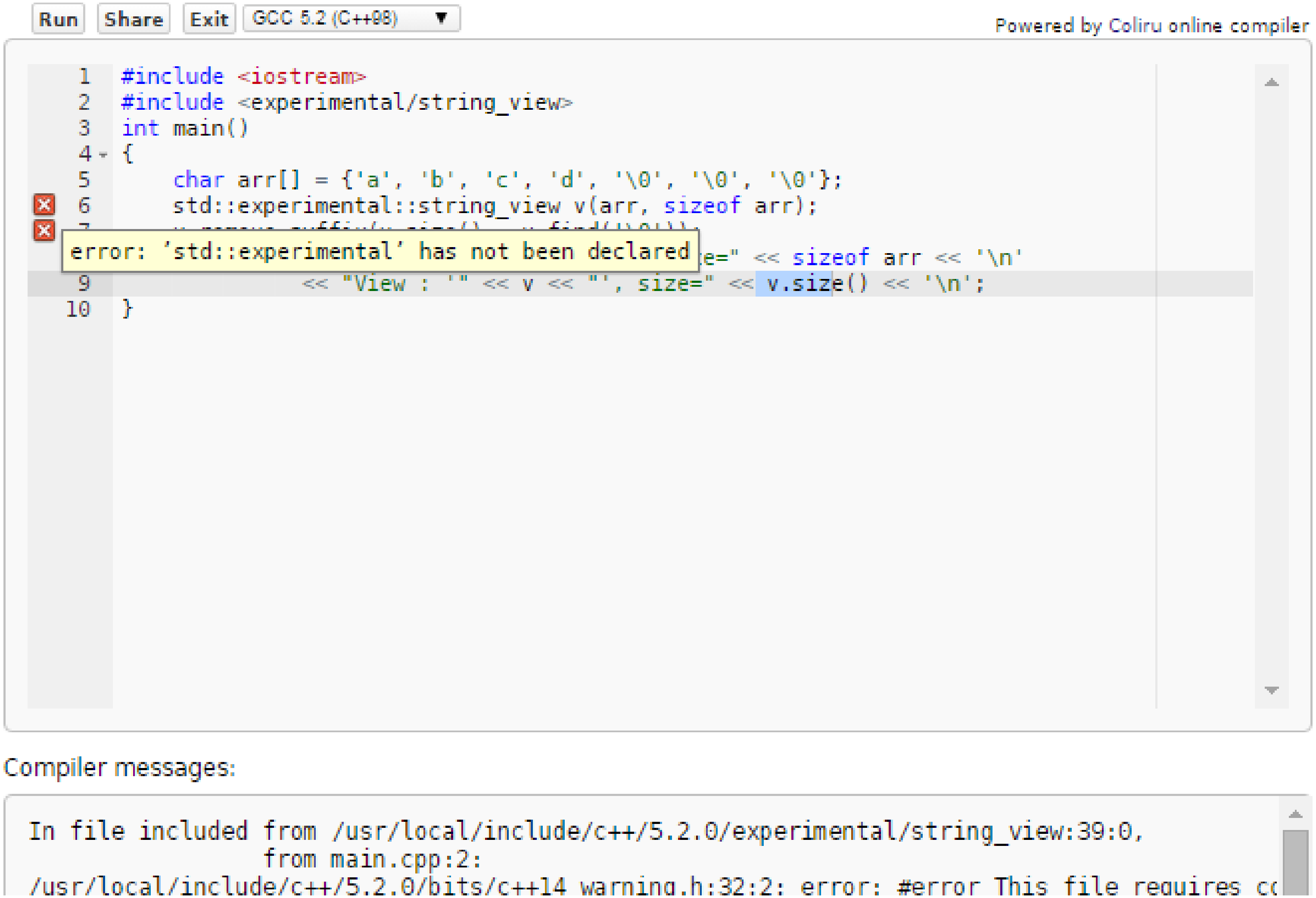
```
#include <
#include <                         ing_view>
int main(
{
    char a                        , 'c', 'd', '\0', '\0', '\0'};
    std::e                        ing_view v(arr, sizeof arr);
    v.remo                        () - v.find('\0'));
    std::c                        ' << arr << "', size=" << sizeof arr << '\n'
                                  ' << v << "', size=" << v.size() << '\n';
}
```



- (cur | prev) ○ 13:26, 20 August 2013 P12 (Talk | contribs | block) . . (6,642 bytes) **(+6,382)** . . *(paste some code from coliru hoping StackedCrooked won't mind)* (undo)
- (cur | prev) ○ 12:35, 20 August 2013 P12 (Talk | contribs | block) . . (260 bytes) (-52) . . (undo)
- (cur | prev) ○ 12:24, 20 August 2013 P12 (Talk | contribs | block) . . (312 bytes) (+312) . . *(let's see if this does anything)*

Inline compiler

Example

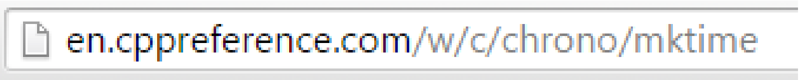


```
#include <iostream>
#include <experimental/string_view>
int main()
{
    char arr[] = {'a', 'b', 'c', 'd', '\0', '\0', '\0'};
    std::experimental::string_view v(arr, sizeof arr);
    v.remove_suffix(v.size() - v.find('\0'));
error: 'std::experimental' has not been declared e=" << sizeof arr << '\n'
              << "View : '" << v << "', size=" << v.size() << '\n';
}
```

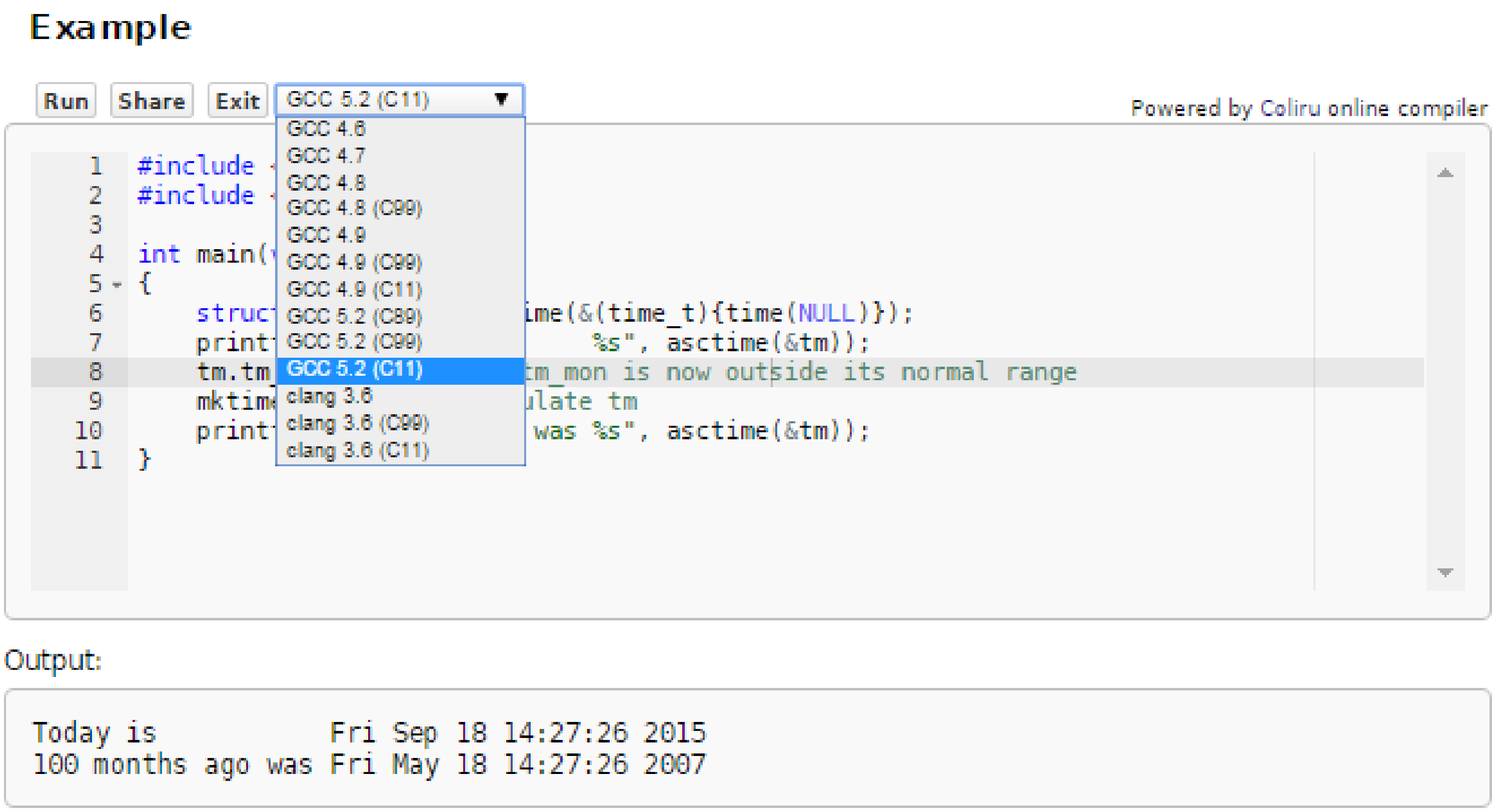
Compiler messages:

```
In file included from /usr/local/include/c++/5.2.0/experimental/string_view:39:0,
                 from main.cpp:2:
/usr/local/include/c++/5.2.0/bits/c++14_warning.h:32:2: error: #error This file requires cc
```

Inline compiler



en.cppreference.com/w/c/chrono/mktime



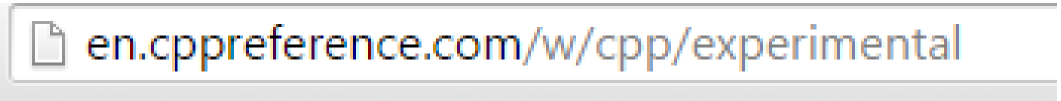
Example

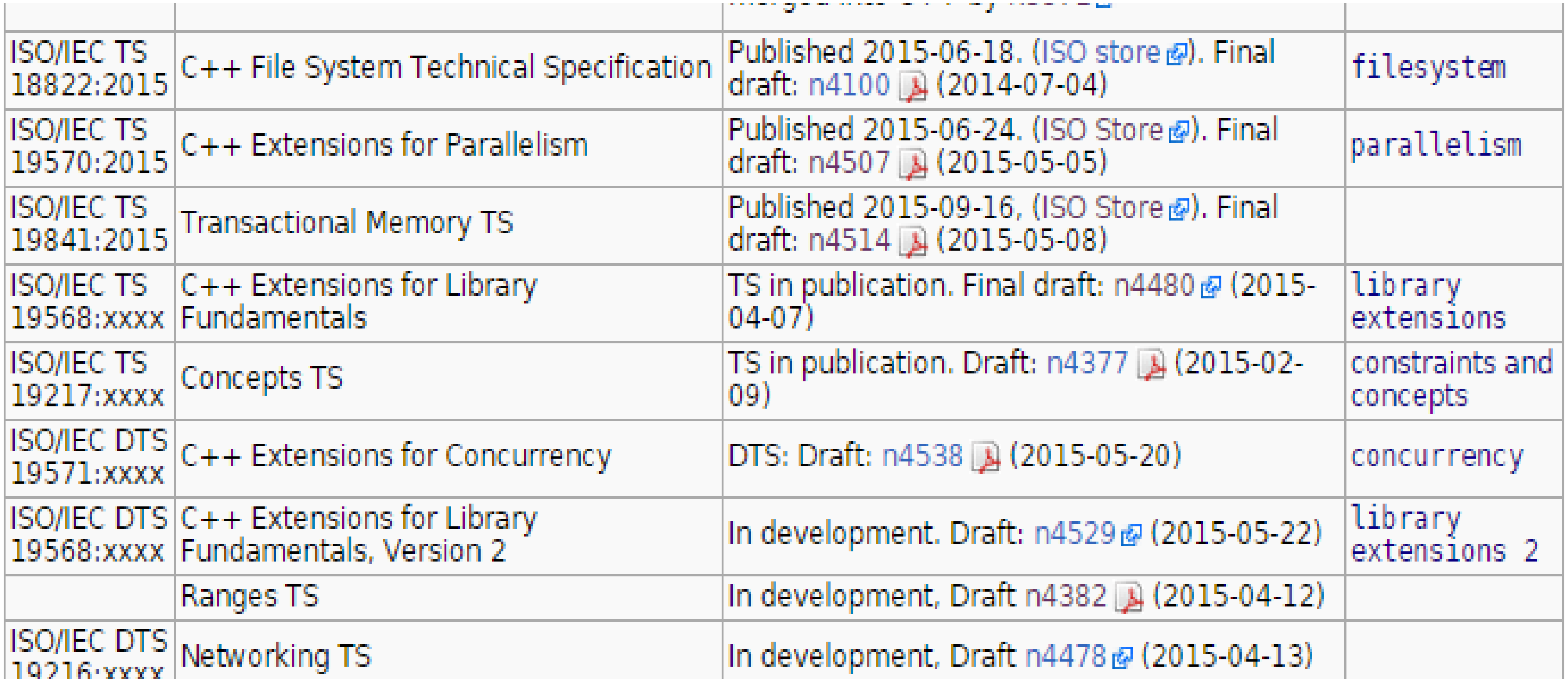
```
#include
#include

int main(
{
    struc                   ime(&(time_t){time(NULL)});
    print                        %s", asctime(&tm));
    tm.tm                   tm_mon is now outside its normal range
    mktim                   ulate tm
    print                   was %s", asctime(&tm));
}
```

Output:

```
Today is           Fri Sep 18 14:27:26 2015
100 months ago was Fri May 18 14:27:26 2007
```

Technical specifications: C++

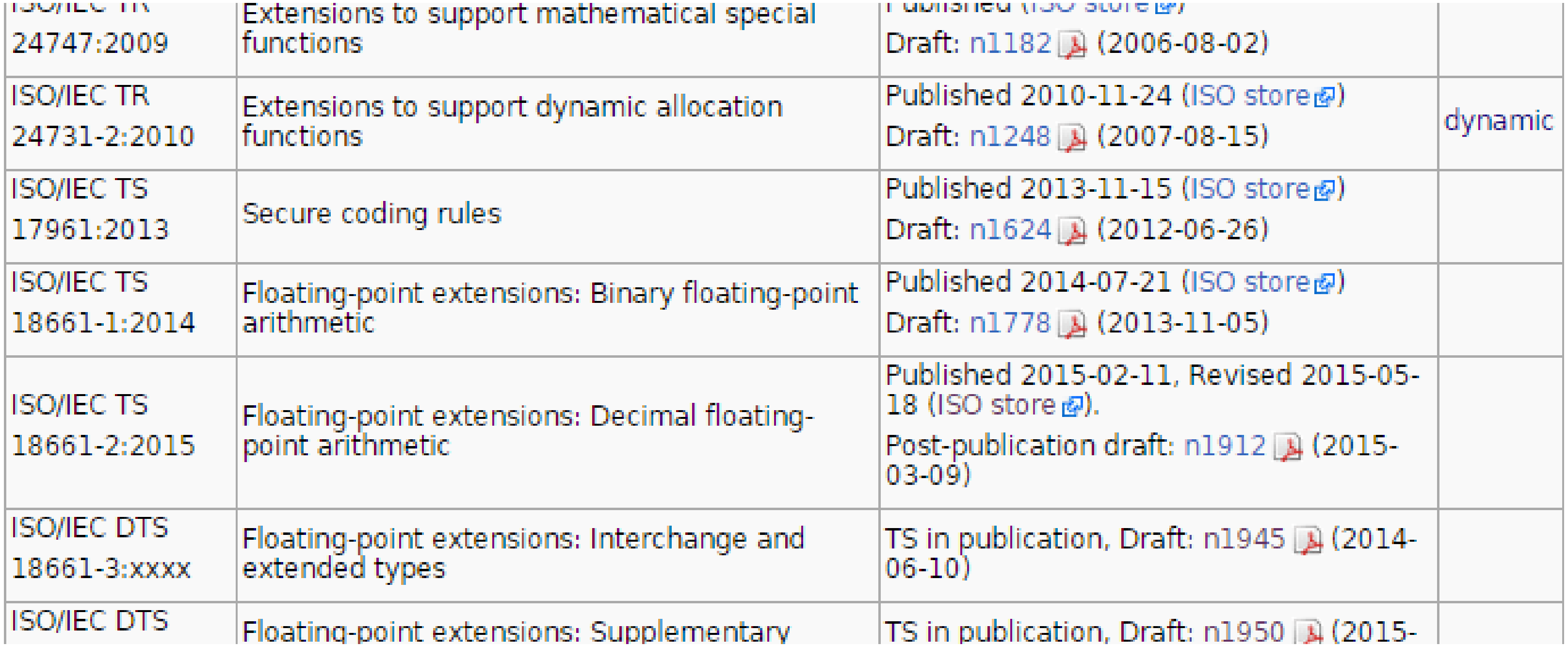

en.cppreference.com/w/cpp/experimental



ISO/IEC TS 18822:2015	C++ File System Technical Specification	Published 2015-06-18. (ISO store). Final draft: n4100 (2014-07-04)	filesystem
ISO/IEC TS 19570:2015	C++ Extensions for Parallelism	Published 2015-06-24. (ISO Store). Final draft: n4507 (2015-05-05)	parallelism
ISO/IEC TS 19841:2015	Transactional Memory TS	Published 2015-09-16, (ISO Store). Final draft: n4514 (2015-05-08)	
ISO/IEC TS 19568:xxxx	C++ Extensions for Library Fundamentals	TS in publication. Final draft: n4480 (2015-04-07)	library extensions
ISO/IEC TS 19217:xxxx	Concepts TS	TS in publication. Draft: n4377 (2015-02-09)	constraints and concepts
ISO/IEC DTS 19571:xxxx	C++ Extensions for Concurrency	DTS: Draft: n4538 (2015-05-20)	concurrency
ISO/IEC DTS 19568:xxxx	C++ Extensions for Library Fundamentals, Version 2	In development. Draft: n4529 (2015-05-22)	library extensions 2
	Ranges TS	In development, Draft n4382 (2015-04-12)	
ISO/IEC DTS 19216:xxxx	Networking TS	In development, Draft n4478 (2015-04-13)	

Technical specifications: C

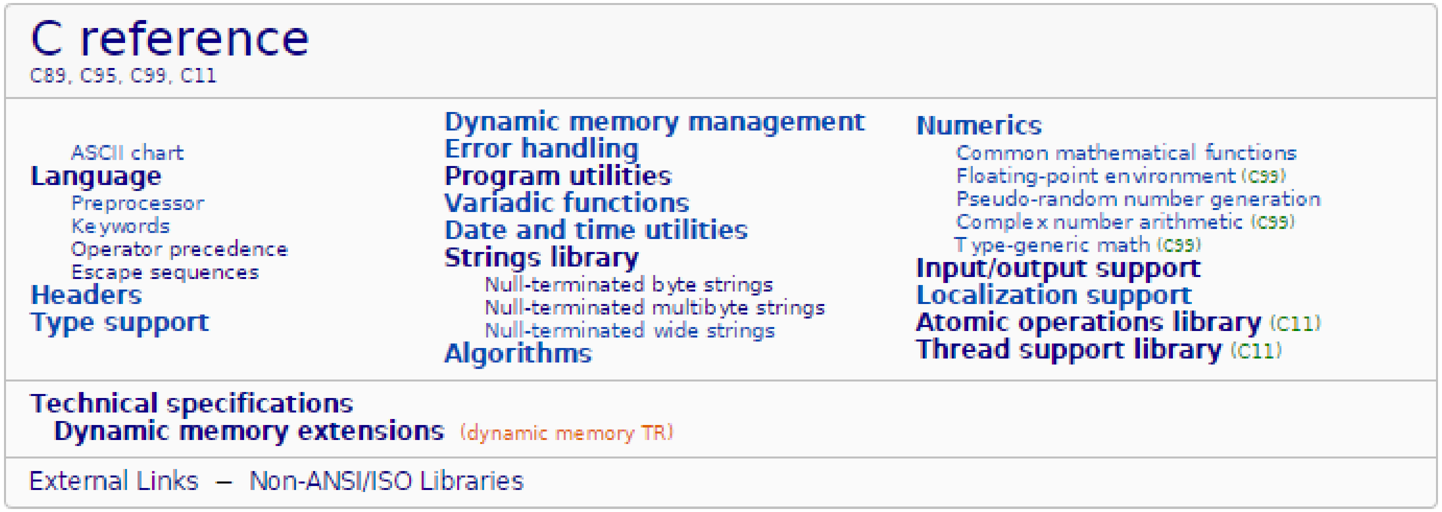
en.cppreference.com/w/c/experimental



ISO/IEC TR 24747:2009	Extensions to support mathematical special functions	Published (ISO store) Draft: n1182 (2006-08-02)	
ISO/IEC TR 24731-2:2010	Extensions to support dynamic allocation functions	Published 2010-11-24 (ISO store) Draft: n1248 (2007-08-15)	dynamic
ISO/IEC TS 17961:2013	Secure coding rules	Published 2013-11-15 (ISO store) Draft: n1624 (2012-06-26)	
ISO/IEC TS 18661-1:2014	Floating-point extensions: Binary floating-point arithmetic	Published 2014-07-21 (ISO store) Draft: n1778 (2013-11-05)	
ISO/IEC TS 18661-2:2015	Floating-point extensions: Decimal floating-point arithmetic	Published 2015-02-11, Revised 2015-05-18 (ISO store). Post-publication draft: n1912 (2015-03-09)	
ISO/IEC DTS 18661-3:xxxx	Floating-point extensions: Interchange and extended types	TS in publication, Draft: n1945 (2014-06-10)	
ISO/IEC DTS	Floating-point extensions: Supplementary	TS in publication, Draft: n1950 (2015-	

The (only?) modern C reference

- C11 library and core language (even annex L)
- Post-C11 defect reports
- ISO Technical Specifications
- History and libraries



C reference

C89, C95, C99, C11

ASCII chart

Language

Preprocessor

Keywords

Operator precedence

Escape sequences

Headers

Type support

Dynamic memory management

Error handling

Program utilities

Variadic functions

Date and time utilities

Strings library

Null-terminated byte strings

Null-terminated multibyte strings

Null-terminated wide strings

Algorithms

Numerics

Common mathematical functions

Floating-point environment (C99)

Pseudo-random number generation

Complex number arithmetic (C99)

Type-generic math (C99)

Input/output support

Localization support

Atomic operations library (C11)

Thread support library (C11)

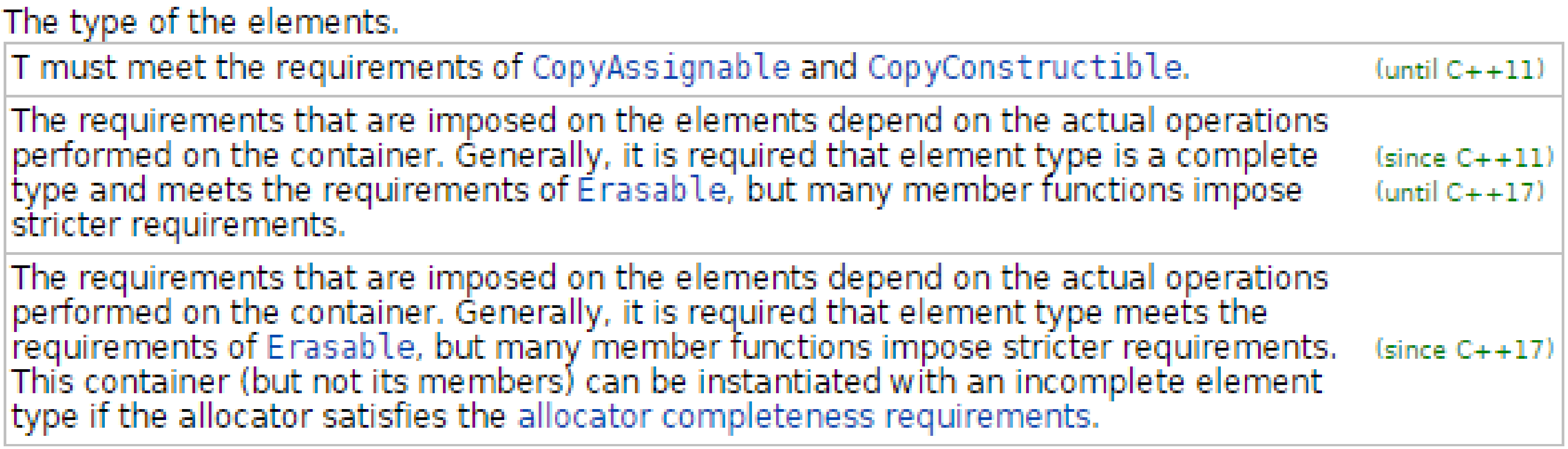
Technical specifications

Dynamic memory extensions (dynamic memory TR)

External Links – Non-ANSI/ISO Libraries

Revision control

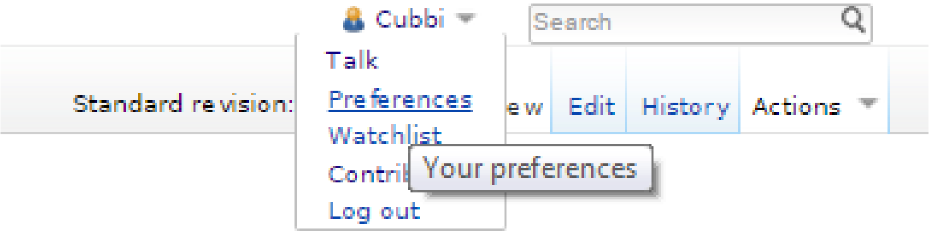
en.cppreference.com/w/cpp/container/vector



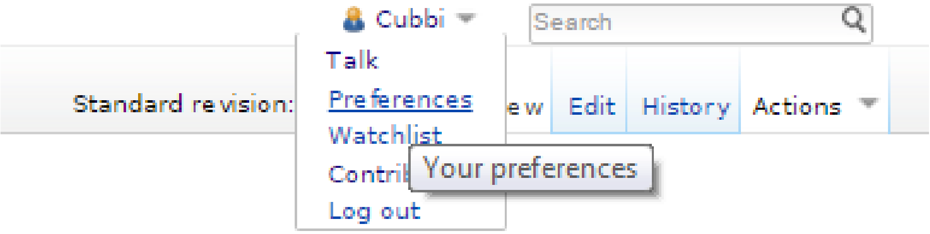
The type of the elements.

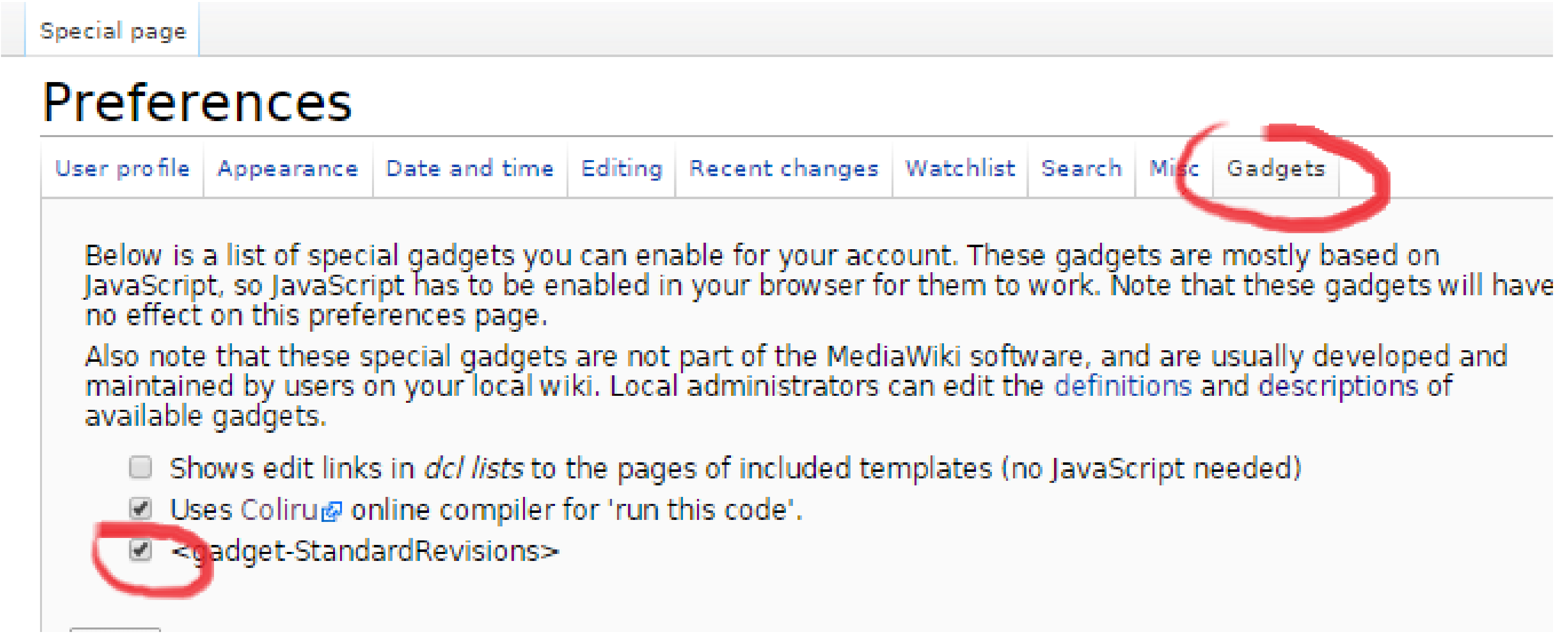
T must meet the requirements of CopyAssignable and CopyConstructible.	(until C++11)
The requirements that are imposed on the elements depend on the actual operations performed on the container. Generally, it is required that element type is a complete type and meets the requirements of Erasable, but many member functions impose stricter requirements.	(since C++11) (until C++17)
The requirements that are imposed on the elements depend on the actual operations performed on the container. Generally, it is required that element type meets the requirements of Erasable, but many member functions impose stricter requirements. This container (but not its members) can be instantiated with an incomplete element type if the allocator satisfies the allocator completeness requirements.	(since C++17)

Revision control



Revision control





Preferences

Below is a list of special gadgets you can enable for your account. These gadgets are mostly based on JavaScript, so JavaScript has to be enabled in your browser for them to work. Note that these gadgets will have no effect on this preferences page.

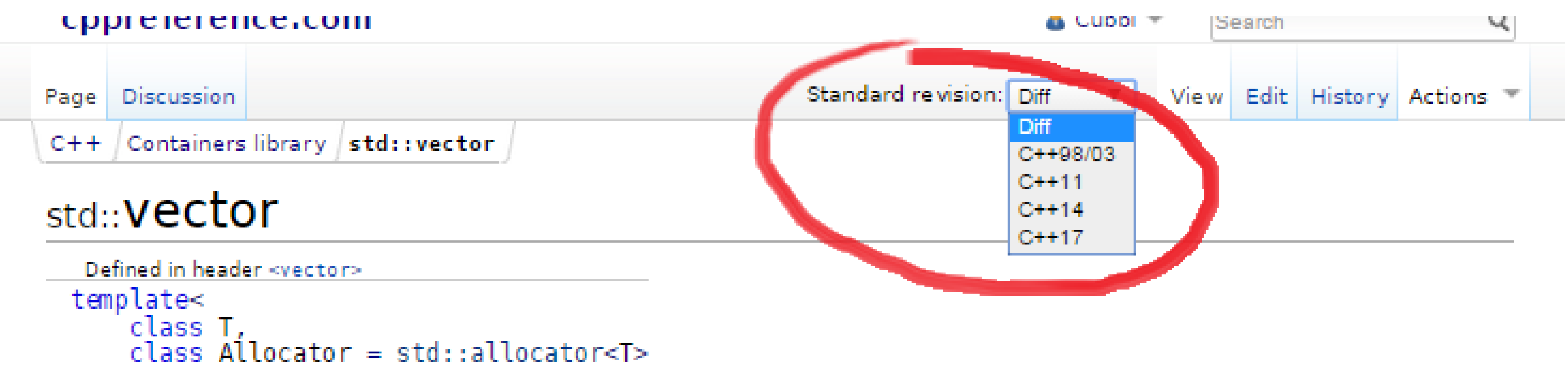
Also note that these special gadgets are not part of the MediaWiki software, and are usually developed and maintained by users on your local wiki. Local administrators can edit the definitions and descriptions of available gadgets.

☐ Shows edit links in *dcl lists* to the pages of included templates (no JavaScript needed)

☑ Uses Coliru online compiler for 'run this code'.

☑ <gadget-StandardRevisions>

Revision control



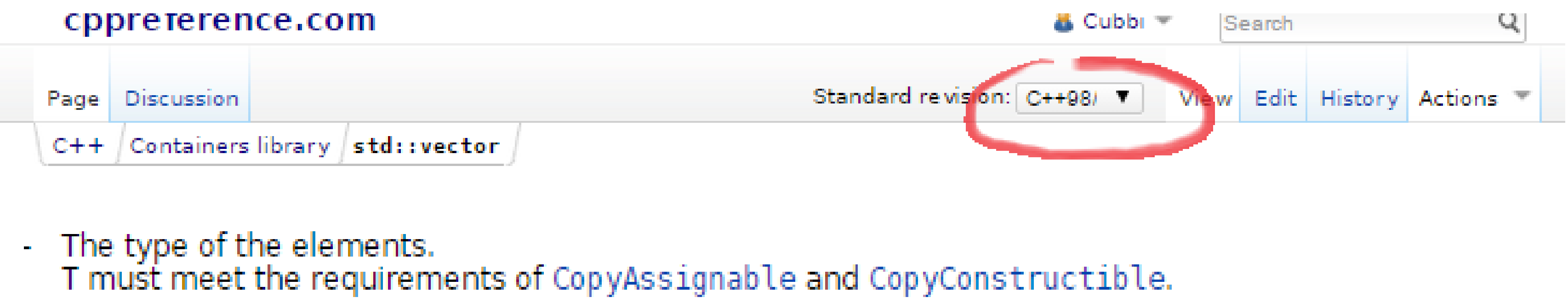
Standard revision: Diff

std::vector

Defined in header <vector>

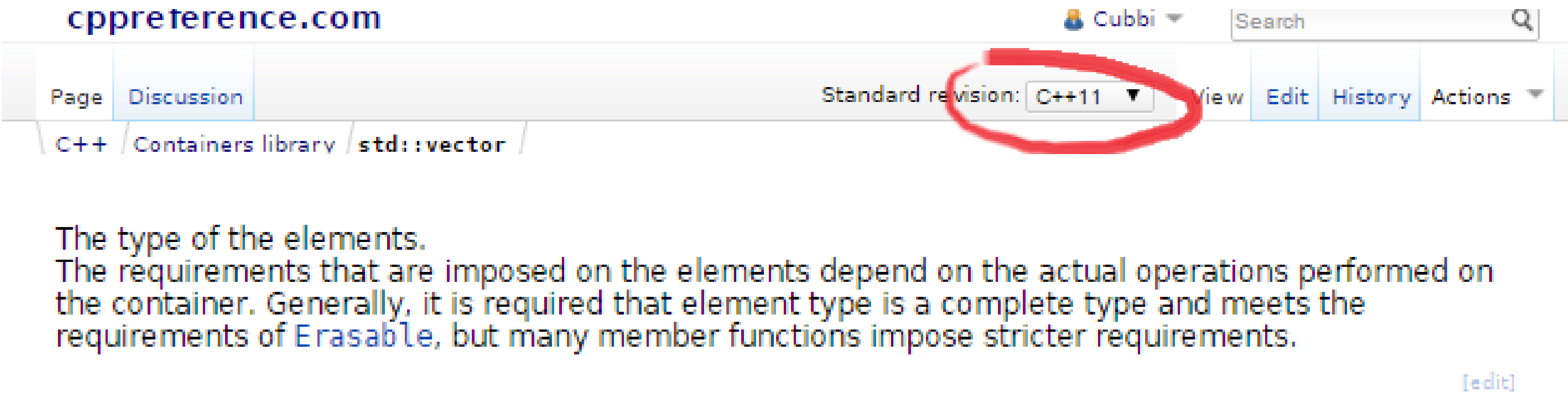
```
template<
    class T,
    class Allocator = std::allocator<T>
```

Revision control



Standard revision: C++98/

- The type of the elements.
  T must meet the requirements of CopyAssignable and CopyConstructible.

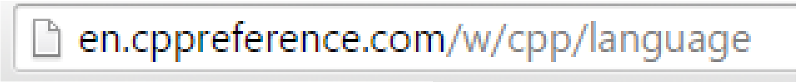


Standard revision: C++11

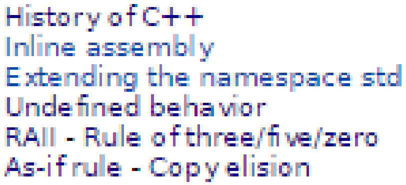
The type of the elements.
The requirements that are imposed on the elements depend on the actual operations performed on the container. Generally, it is required that element type is a complete type and meets the requirements of Erasable, but many member functions impose stricter requirements.

Beyond the spec

- Idioms and patterns

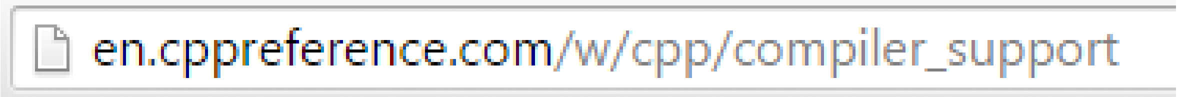


en.cppreference.com/w/cpp/language



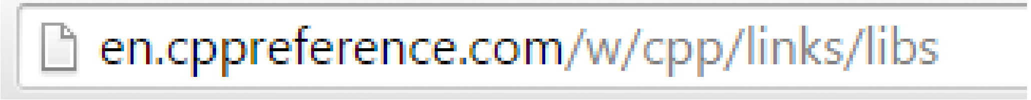
History of C++
Inline assembly
Extending the namespace std
Undefined behavior
RAII - Rule of three/five/zero
As-if rule - Copy elision

- The state of compiler support



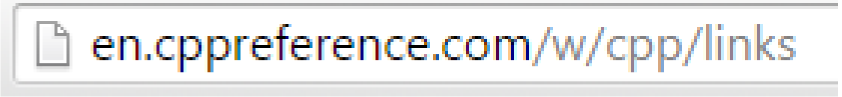
en.cppreference.com/w/cpp/compiler_support

- Third-party libraries



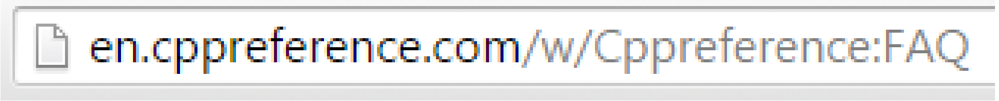
en.cppreference.com/w/cpp/links/libs

- Other C++ references

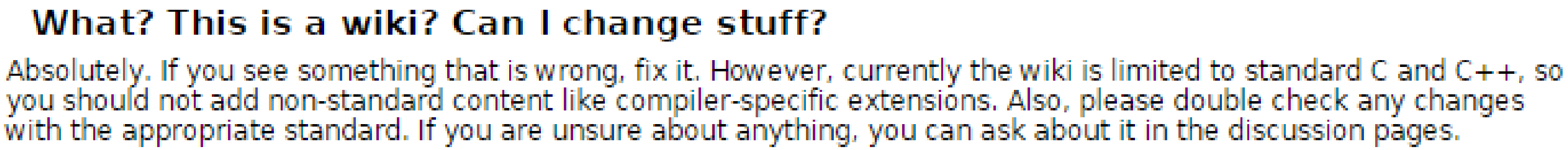


en.cppreference.com/w/cpp/links

It’s a wiki, you can change stuff!



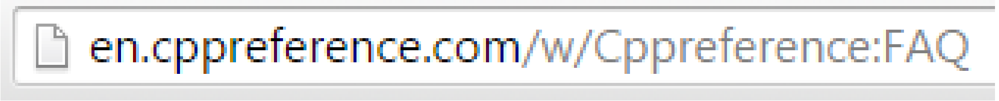
en.cppreference.com/w/Cppreference:FAQ



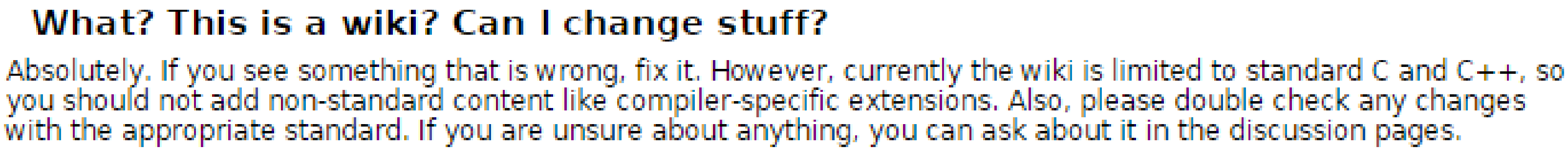
What? This is a wiki? Can I change stuff?

Absolutely. If you see something that is wrong, fix it. However, currently the wiki is limited to standard C and C++, so you should not add non-standard content like compiler-specific extensions. Also, please double check any changes with the appropriate standard. If you are unsure about anything, you can ask about it in the discussion pages.

It’s a wiki, you can change stuff!



en.cppreference.com/w/Cppreference:FAQ



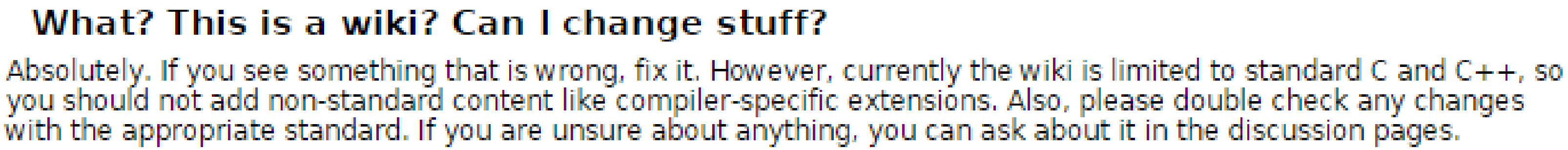
What? This is a wiki? Can I change stuff?

Absolutely. If you see something that is wrong, fix it. However, currently the wiki is limited to standard C and C++, so you should not add non-standard content like compiler-specific extensions. Also, please double check any changes with the appropriate standard. If you are unsure about anything, you can ask about it in the discussion pages.

Easier said than done…

It’s a wiki, you can change stuff!

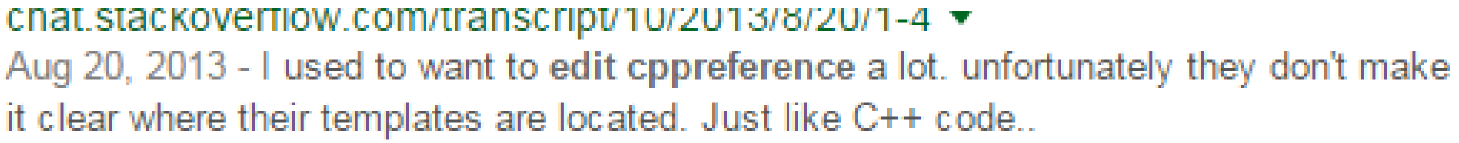
en.cppreference.com/w/Cppreference:FAQ



What? This is a wiki? Can I change stuff?

Absolutely. If you see something that is wrong, fix it. However, currently the wiki is limited to standard C and C++, so you should not add non-standard content like compiler-specific extensions. Also, please double check any changes with the appropriate standard. If you are unsure about anything, you can ask about it in the discussion pages.

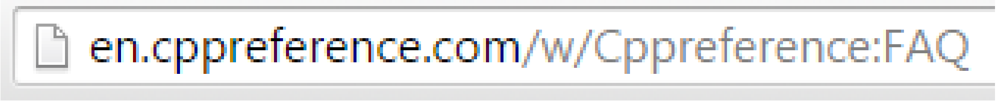
Easier said than done…

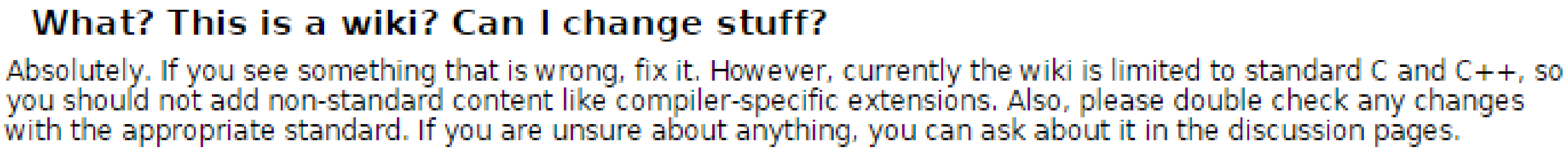


chat.stackoverflow.com/transcript/10/2013/8/20/1-4 ▾

Aug 20, 2013 - I used to want to **edit cppreference** a lot. unfortunately they don't make it clear where their templates are located. Just like C++ code..

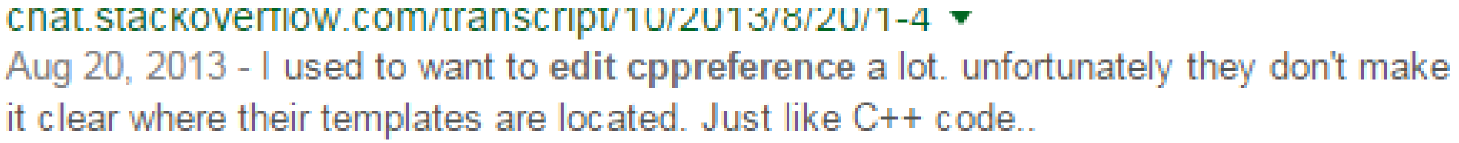
It’s a wiki, you can change stuff!


en.cppreference.com/w/Cppreference:FAQ


What? This is a wiki? Can I change stuff?

Absolutely. If you see something that is wrong, fix it. However, currently the wiki is limited to standard C and C++, so you should not add non-standard content like compiler-specific extensions. Also, please double check any changes with the appropriate standard. If you are unsure about anything, you can ask about it in the discussion pages.

Easier said than done…

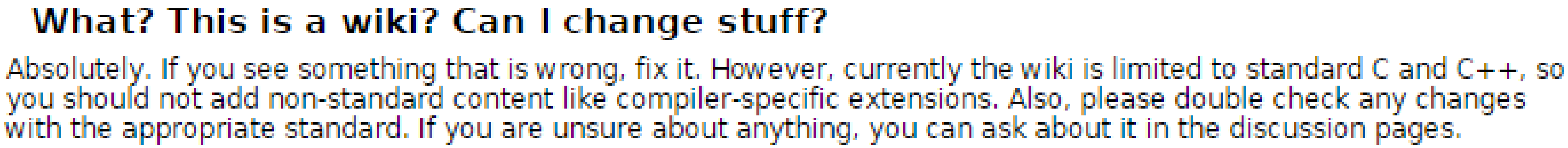

chat.stackoverflow.com/transcript/10/2013/8/20/1-4 ▾

Aug 20, 2013 - I used to want to **edit cppreference** a lot. unfortunately they don't make it clear where their templates are located. Just like C++ code..

Just like C++, **cppreference** is riddled with templates.

It’s a wiki, you can change stuff!

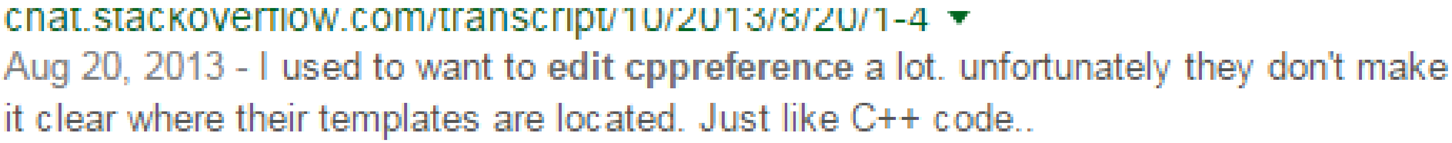
en.cppreference.com/w/Cppreference:FAQ



What? This is a wiki? Can I change stuff?

Absolutely. If you see something that is wrong, fix it. However, currently the wiki is limited to standard C and C++, so you should not add non-standard content like compiler-specific extensions. Also, please double check any changes with the appropriate standard. If you are unsure about anything, you can ask about it in the discussion pages.

Easier said than done…



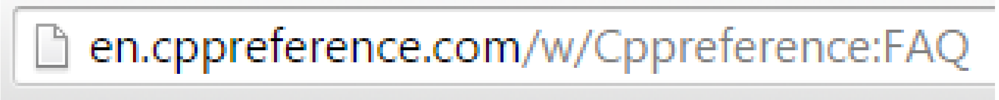
chat.stackoverflow.com/transcript/10/2013/8/20/1-4 ▾

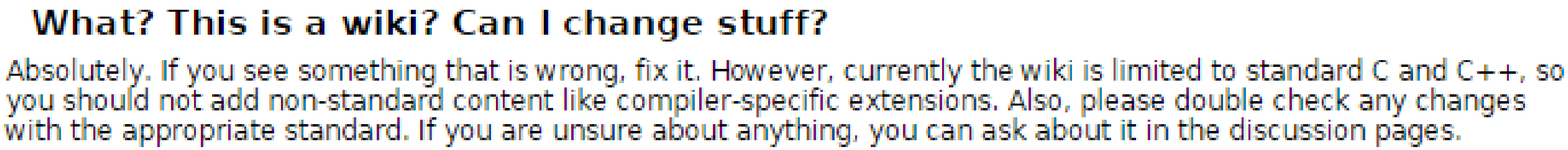
Aug 20, 2013 - I used to want to **edit cppreference** a lot. unfortunately they don't make it clear where their templates are located. Just like C++ code..

Just like C++, **cppreference** is riddled with templates.

. everyone is too scared of **mediawiki** templates to bother tbh.

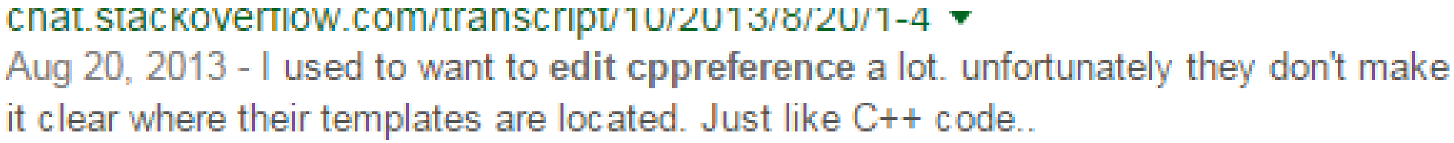
It’s a wiki, you can change stuff!


en.cppreference.com/w/Cppreference:FAQ

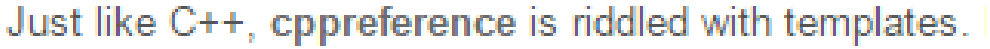

What? This is a wiki? Can I change stuff?

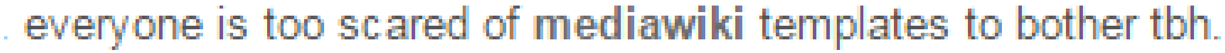
Absolutely. If you see something that is wrong, fix it. However, currently the wiki is limited to standard C and C++, so you should not add non-standard content like compiler-specific extensions. Also, please double check any changes with the appropriate standard. If you are unsure about anything, you can ask about it in the discussion pages.

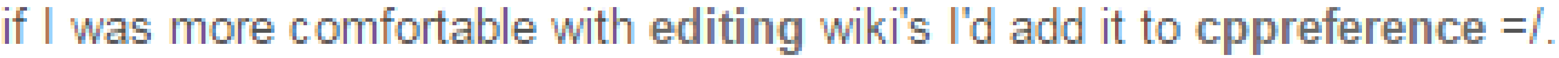
Easier said than done…


chat.stackoverflow.com/transcript/10/2013/8/20/1-4 ▾

Aug 20, 2013 - I used to want to **edit cppreference** a lot. unfortunately they don't make it clear where their templates are located. Just like C++ code..

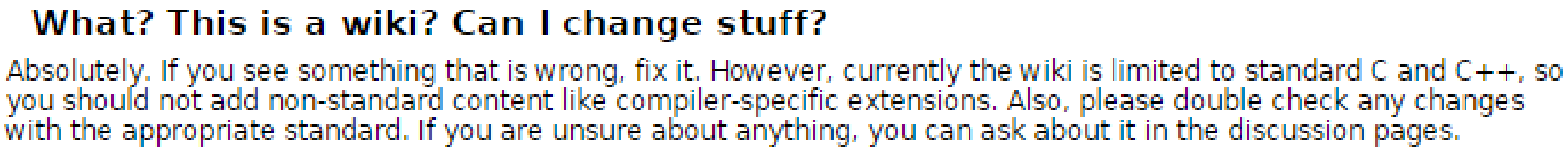

Just like C++, **cppreference** is riddled with templates.


. everyone is too scared of **mediawiki** templates to bother tbh.


if I was more comfortable with **editing** wiki's I'd add it to **cppreference** =/.

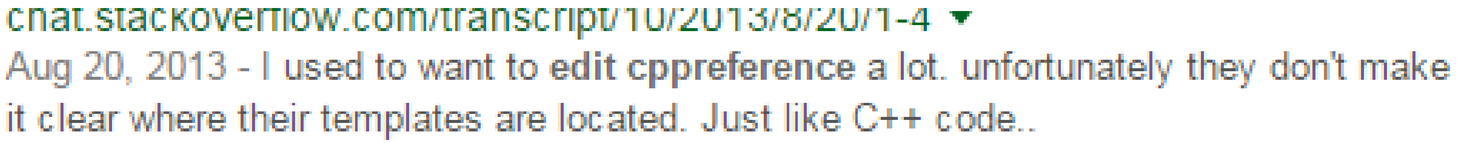
It’s a wiki, you can change stuff!

en.cppreference.com/w/Cppreference:FAQ


What? This is a wiki? Can I change stuff?

Absolutely. If you see something that is wrong, fix it. However, currently the wiki is limited to standard C and C++, so you should not add non-standard content like compiler-specific extensions. Also, please double check any changes with the appropriate standard. If you are unsure about anything, you can ask about it in the discussion pages.

Easier said than done…


chat.stackoverflow.com/transcript/10/2013/8/20/1-4 ▾

Aug 20, 2013 - I used to want to **edit cppreference** a lot. unfortunately they don't make it clear where their templates are located. Just like C++ code..

Just like C++, **cppreference** is riddled with templates.

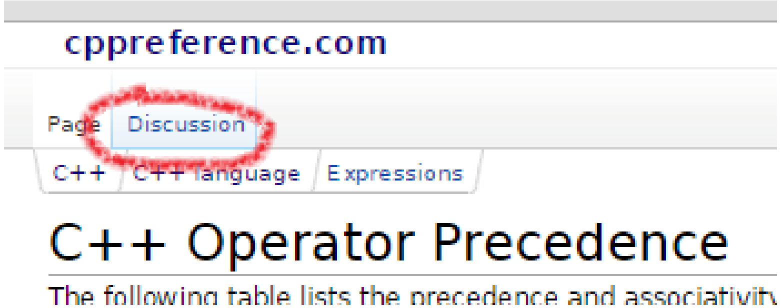
. everyone is too scared of **mediawiki** templates to bother tbh.

if I was more comfortable with **editing** wiki's I'd add it to **cppreference** =/.

May 17, 2012 - @MooingDuck Time to fix **cppreference**.com... Oh gawd ...

If you see something, say something

- Use the Talk (Discussion) page!



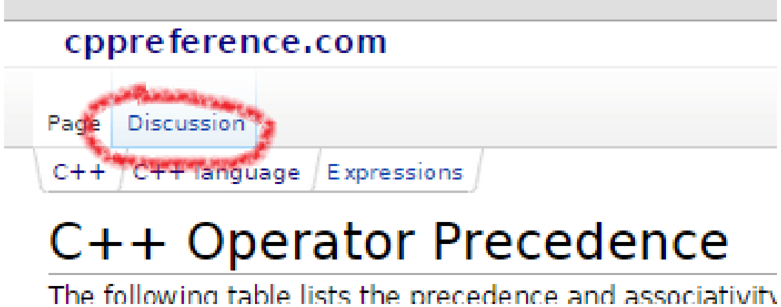
cppreference.com

C++ Operator Precedence

The following table lists the precedence and associativity

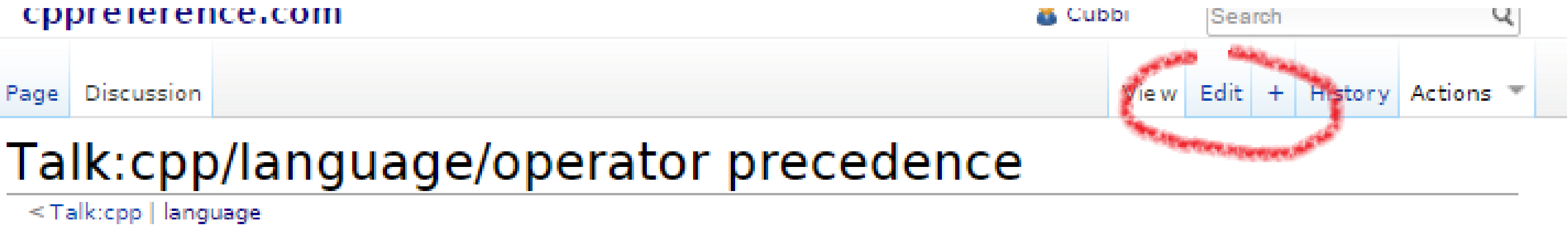
If you see something, say something

- Use the Talk (Discussion) page!



C++ Operator Precedence

The following table lists the precedence and associativity

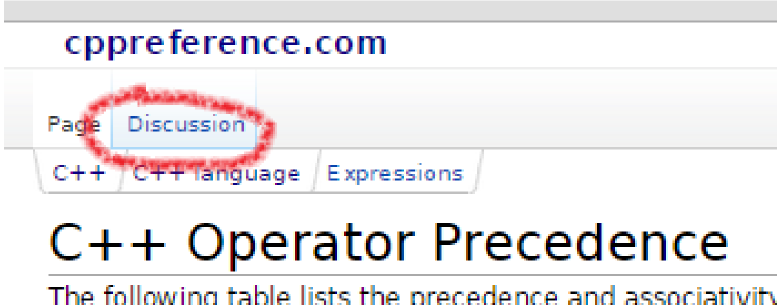


Talk:cpp/language/operator precedence

< Talk:cpp | language

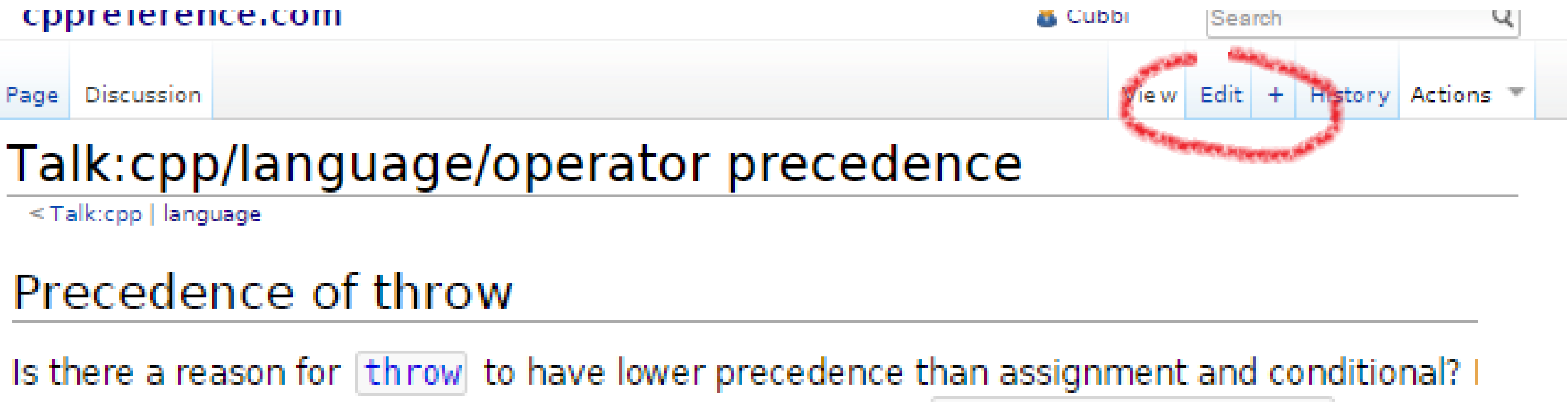
If you see something, say something

- Use the Talk (Discussion) page!



C++ Operator Precedence

The following table lists the precedence and associativity



Talk:cpp/language/operator precedence

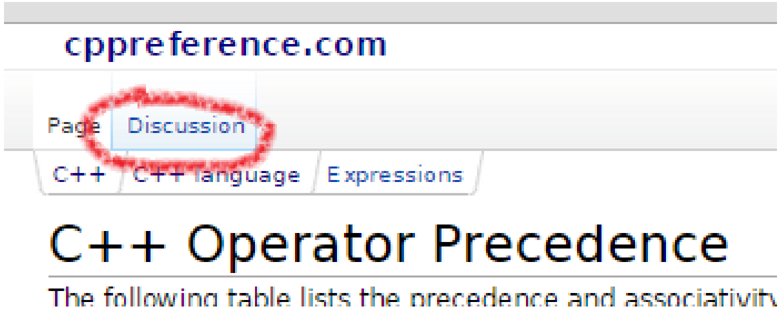
< Talk:cpp | language

Precedence of throw

Is there a reason for `throw` to have lower precedence than assignment and conditional?

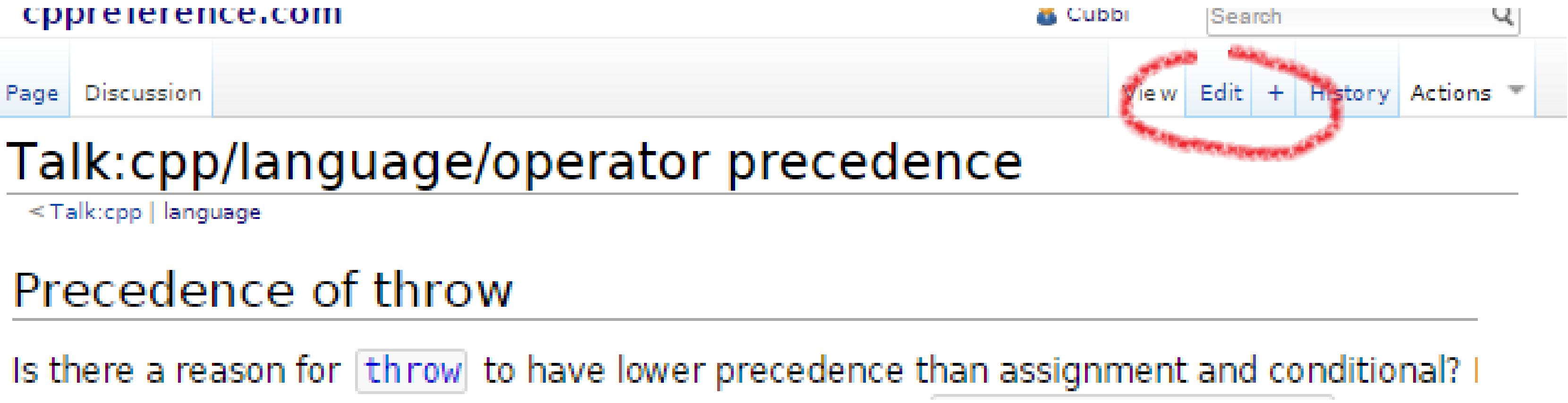
If you see something, say something

- Use the Talk (Discussion) page!



C++ Operator Precedence

The following table lists the precedence and associativity



Talk:cpp/language/operator precedence

< Talk:cpp | language

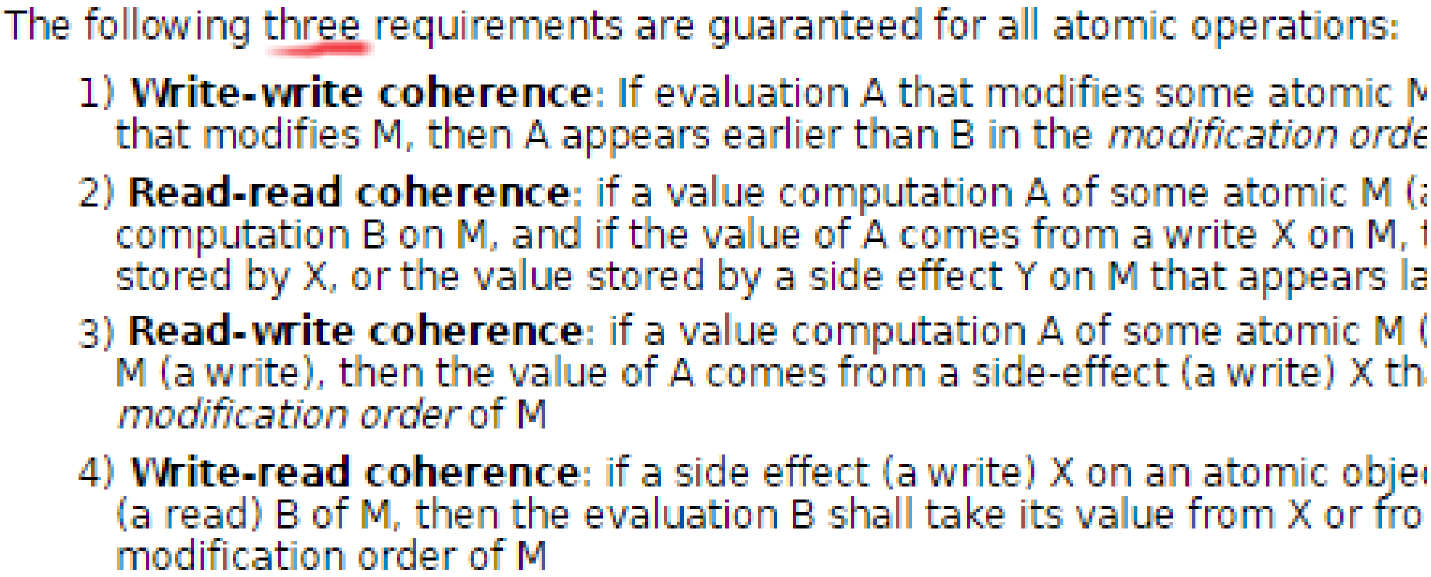
Precedence of throw

Is there a reason for `throw` to have lower precedence than assignment and conditional?

Good job finding a bug in probably the most visited page!

Not everything is a template

en.cppreference.com/w/cpp/atomic/memory_order



The following three requirements are guaranteed for all atomic operations:

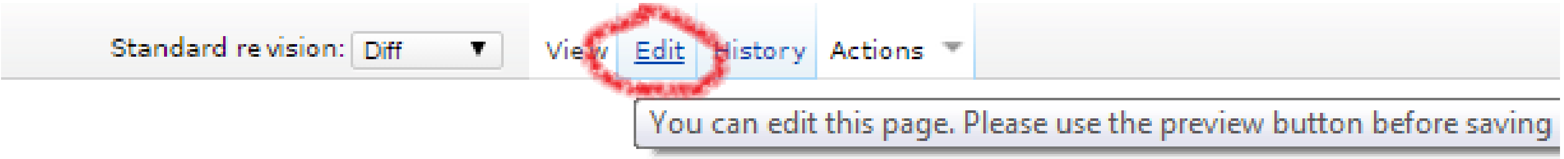
1) **Write-write coherence**: If evaluation A that modifies some atomic M
that modifies M, then A appears earlier than B in the *modification orde*

2) **Read-read coherence**: if a value computation A of some atomic M (a
computation B on M, and if the value of A comes from a write X on M, t
stored by X, or the value stored by a side effect Y on M that appears la

3) **Read-write coherence**: if a value computation A of some atomic M (
M (a write), then the value of A comes from a side-effect (a write) X th
modification order of M

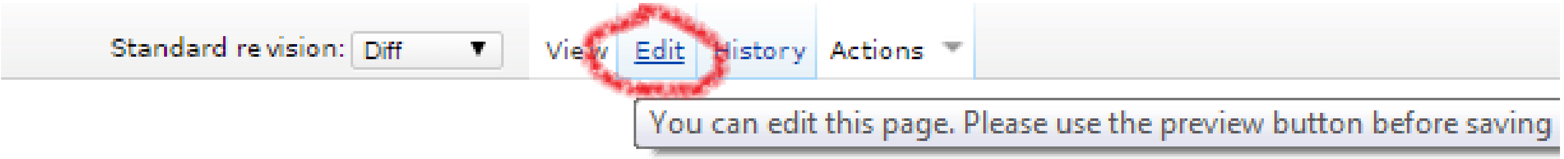
4) **Write-read coherence**: if a side effect (a write) X on an atomic obje
(a read) B of M, then the evaluation B shall take its value from X or fro
modification order of M

Not everything is a template

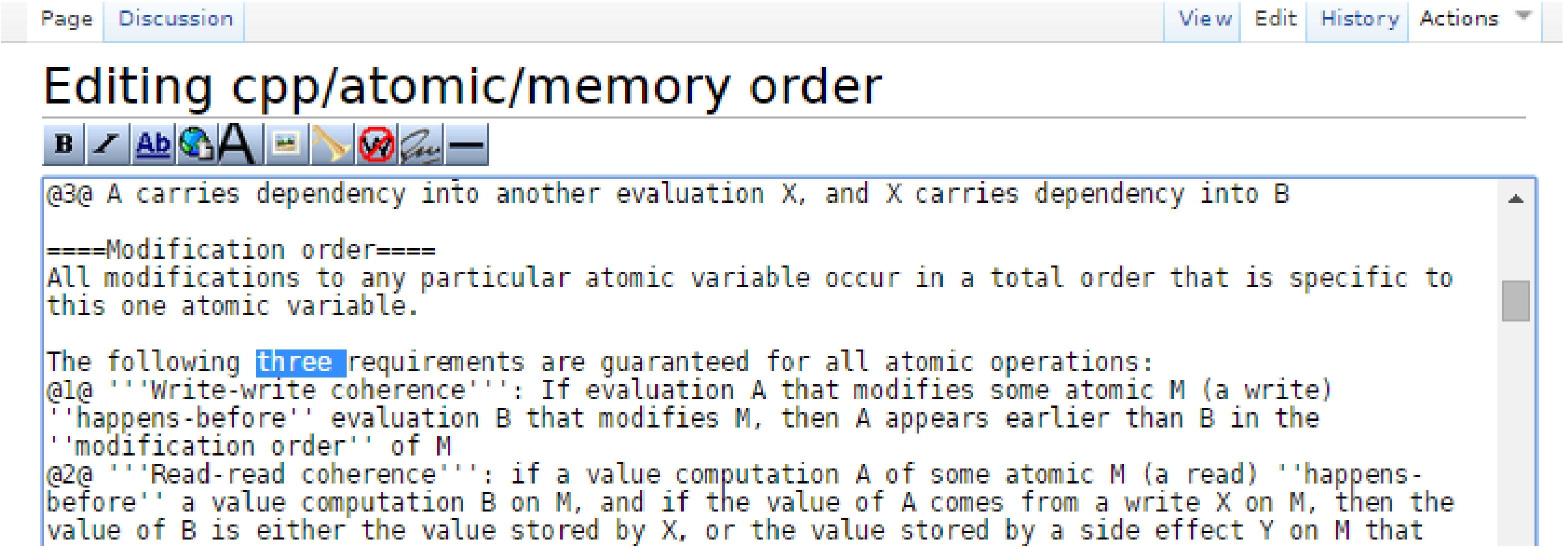


You can edit this page. Please use the preview button before saving

Not everything is a template



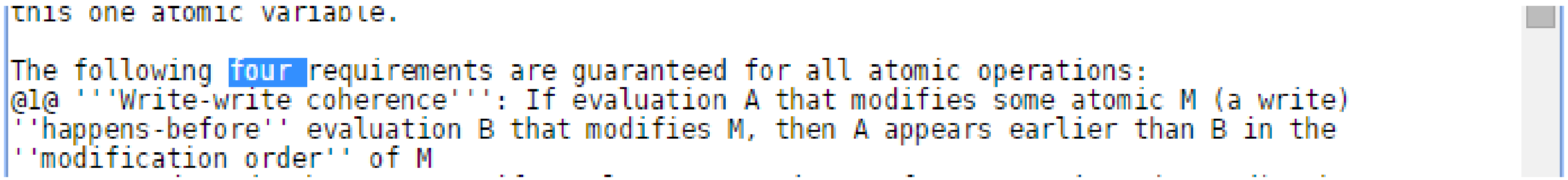
You can edit this page. Please use the preview button before saving



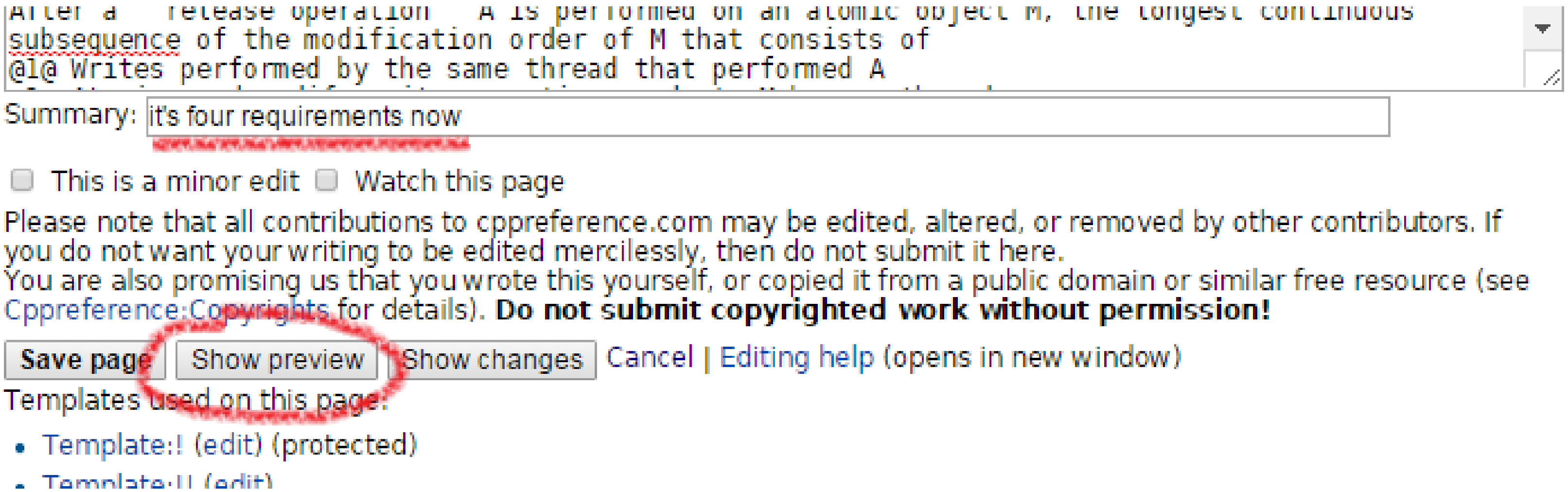
Editing cpp/atomic/memory order

```
@3@ A carries dependency into another evaluation X, and X carries dependency into B

====Modification order====
All modifications to any particular atomic variable occur in a total order that is specific to
this one atomic variable.

The following three requirements are guaranteed for all atomic operations:
@1@ '''Write-write coherence''': If evaluation A that modifies some atomic M (a write)
''happens-before'' evaluation B that modifies M, then A appears earlier than B in the
''modification order'' of M
@2@ '''Read-read coherence''': if a value computation A of some atomic M (a read) ''happens-
before'' a value computation B on M, and if the value of A comes from a write X on M, then the
value of B is either the value stored by X, or the value stored by a side effect Y on M that
```

Not everything is a template



```
this one atomic variable.

The following four requirements are guaranteed for all atomic operations:
@1@ '''Write-write coherence''': If evaluation A that modifies some atomic M (a write)
''happens-before'' evaluation B that modifies M, then A appears earlier than B in the
''modification order'' of M
```



```
After a ''release operation'' A is performed on an atomic object M, the longest continuous
subsequence of the modification order of M that consists of
@1@ Writes performed by the same thread that performed A
```

Summary: it's four requirements now

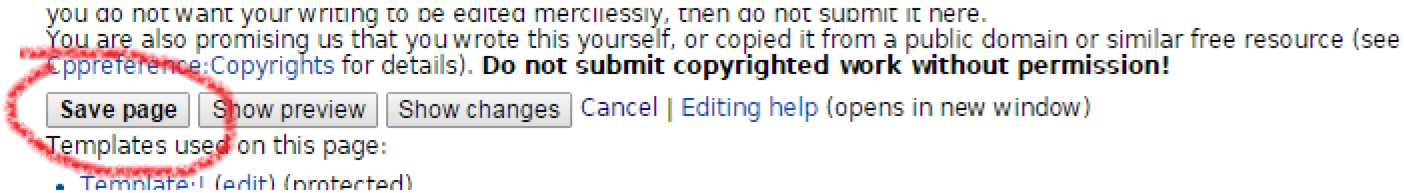
☐ This is a minor edit ☐ Watch this page

Please note that all contributions to cppreference.com may be edited, altered, or removed by other contributors. If you do not want your writing to be edited mercilessly, then do not submit it here.
You are also promising us that you wrote this yourself, or copied it from a public domain or similar free resource (see Cppreference:Copyrights for details). **Do not submit copyrighted work without permission!**

Templates used on this page:

- Template:! (edit) (protected)
- Template:!! (edit)

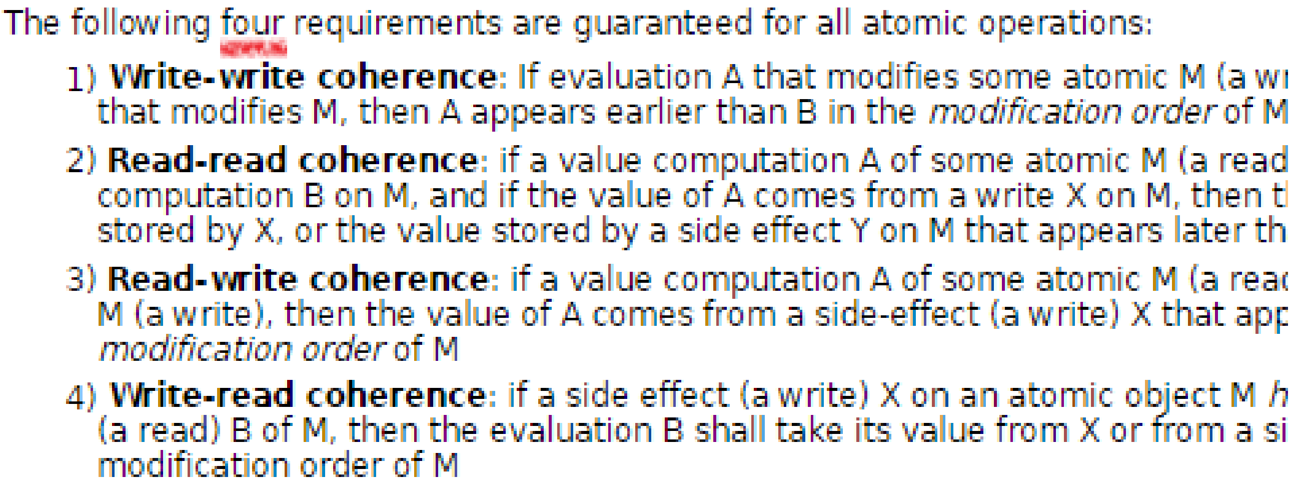
Not everything is a template



you do not want your writing to be edited mercilessly, then do not submit it here.
You are also promising us that you wrote this yourself, or copied it from a public domain or similar free resource (see Cppreference:Copyrights for details). **Do not submit copyrighted work without permission!**

Templates used on this page:

- Template:! (edit) (protected)



The following four requirements are guaranteed for all atomic operations:

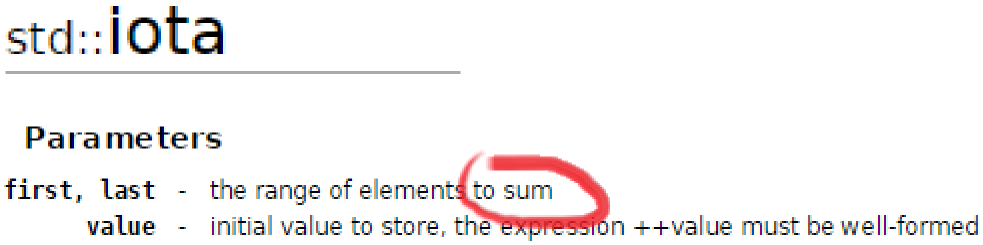
1) **Write-write coherence**: If evaluation A that modifies some atomic M (a wı
that modifies M, then A appears earlier than B in the *modification order* of M

2) **Read-read coherence**: if a value computation A of some atomic M (a read
computation B on M, and if the value of A comes from a write X on M, then tl
stored by X, or the value stored by a side effect Y on M that appears later th

3) **Read-write coherence**: if a value computation A of some atomic M (a reac
M (a write), then the value of A comes from a side-effect (a write) X that apr
modification order of M

4) **Write-read coherence**: if a side effect (a write) X on an atomic object M *h*
(a read) B of M, then the evaluation B shall take its value from X or from a si
modification order of M

Inline templates



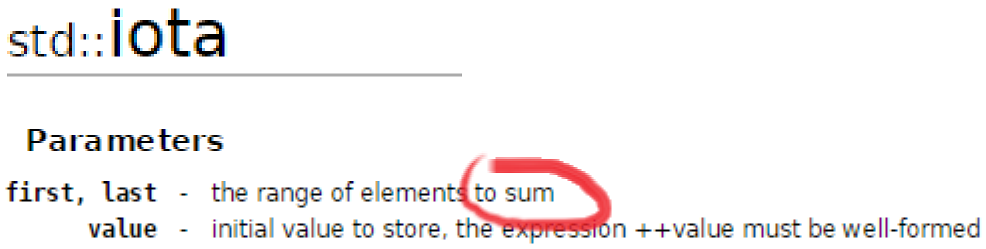
std::iota

Parameters

first, last - the range of elements to sum

value - initial value to store, the expression ++value must be well-formed

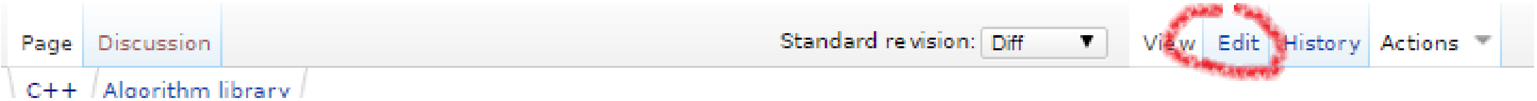
Inline templates



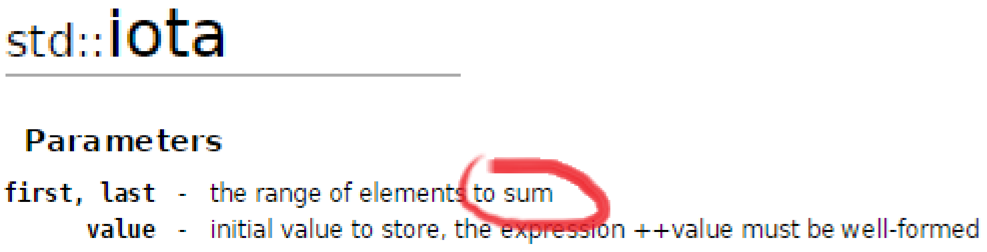
std::iota

Parameters

`first, last`	-	the range of elements to sum
`value`	-	initial value to store, the expression ++value must be well-formed



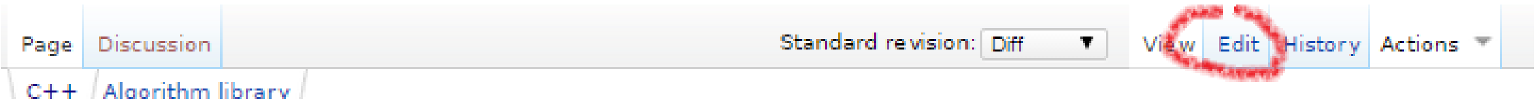
Inline templates

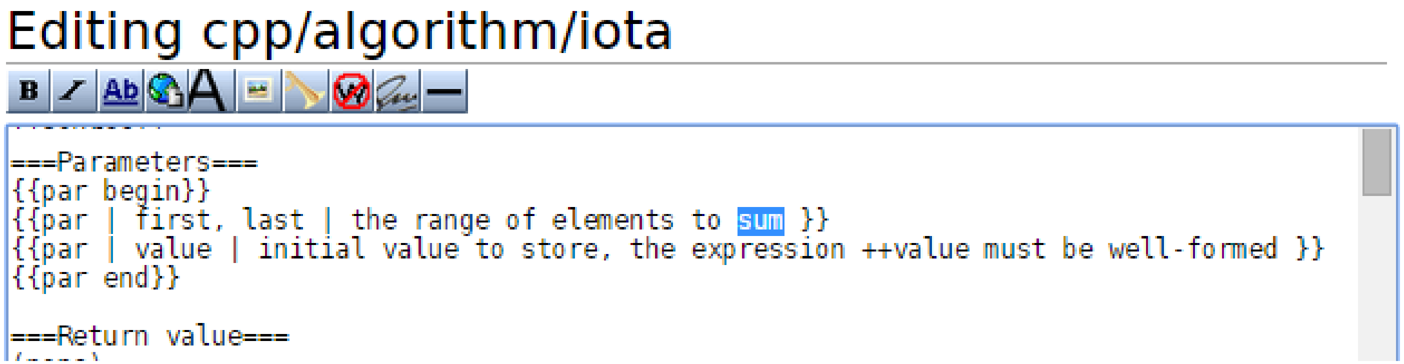


std::iota

Parameters

first, last - the range of elements to sum
value - initial value to store, the expression ++value must be well-formed

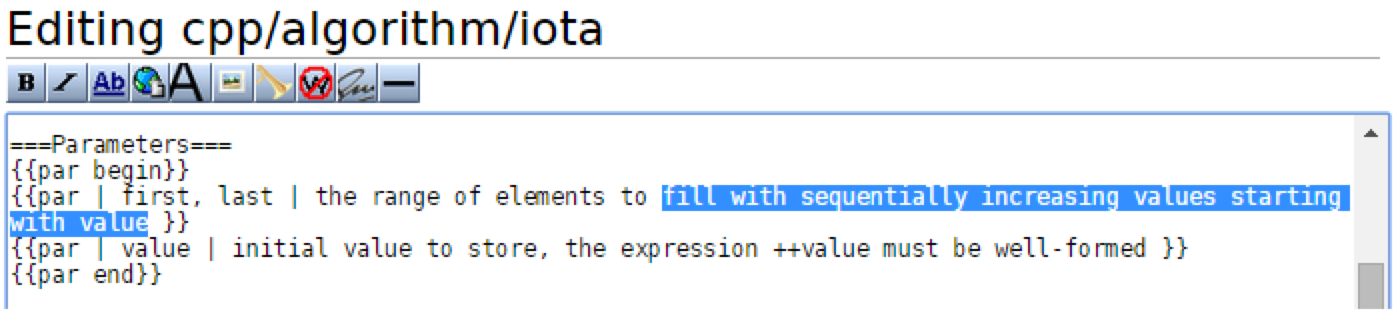




Editing cpp/algorithm/iota

```
===Parameters===
{{par begin}}
{{par | first, last | the range of elements to sum }}
{{par | value | initial value to store, the expression ++value must be well-formed }}
{{par end}}

===Return value===
```

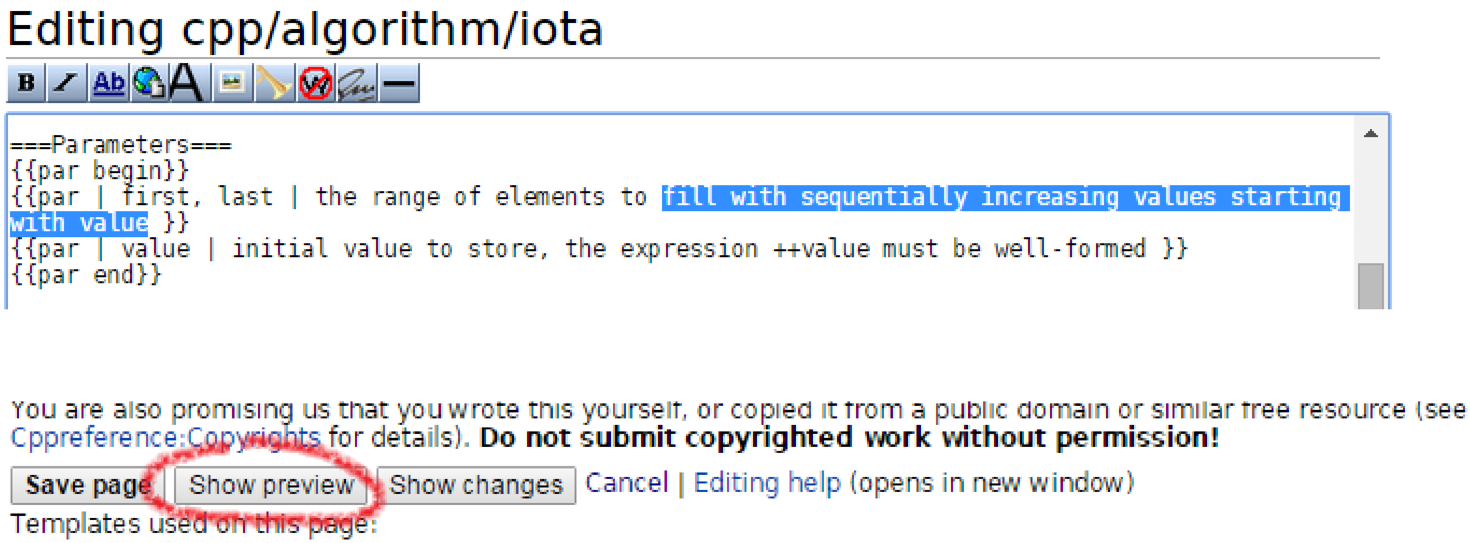
Inline templates



Editing cpp/algorithm/iota

```
===Parameters===
{{par begin}}
{{par | first, last | the range of elements to fill with sequentially increasing values starting with value }}
{{par | value | initial value to store, the expression ++value must be well-formed }}
{{par end}}
```

Inline templates



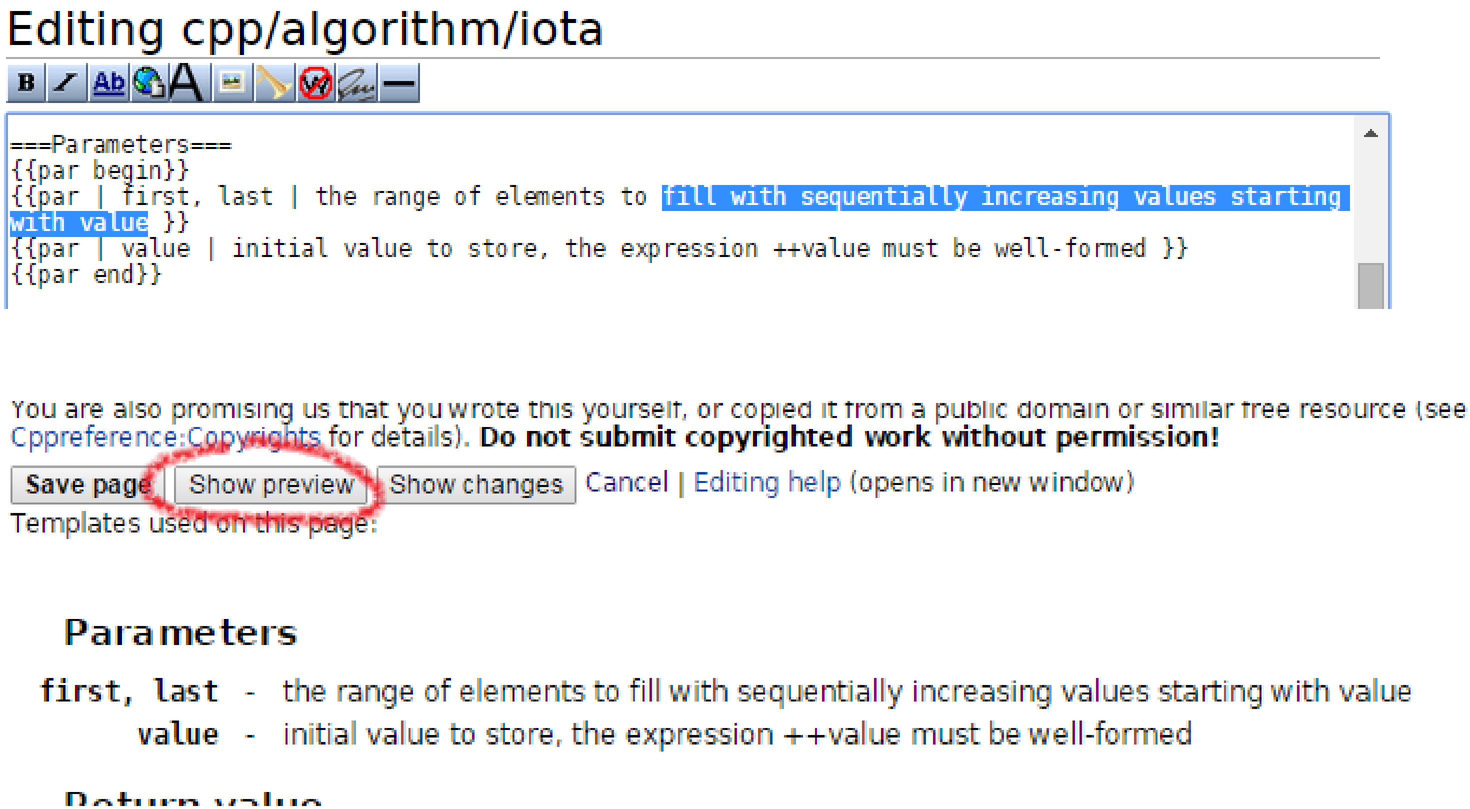
Editing cpp/algorithm/iota

```
===Parameters===
{{par begin}}
{{par | first, last | the range of elements to fill with sequentially increasing values starting
with value }}
{{par | value | initial value to store, the expression ++value must be well-formed }}
{{par end}}
```

You are also promising us that you wrote this yourself, or copied it from a public domain or similar free resource (see Cppreference:Copyrights for details). **Do not submit copyrighted work without permission!**

Templates used on this page:

Inline templates



Editing cpp/algorithm/iota

```
===Parameters===
{{par begin}}
{{par | first, last | the range of elements to fill with sequentially increasing values starting with value }}
{{par | value | initial value to store, the expression ++value must be well-formed }}
{{par end}}
```

You are also promising us that you wrote this yourself, or copied it from a public domain or similar free resource (see Cppreference:Copyrights for details). **Do not submit copyrighted work without permission!**

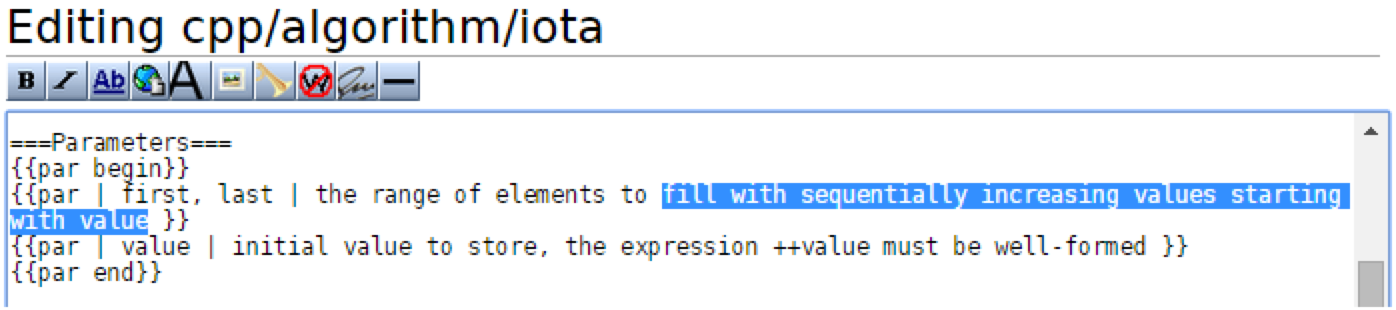
Templates used on this page:

Parameters

first, last	-	the range of elements to fill with sequentially increasing values starting with value
value	-	initial value to store, the expression ++value must be well-formed

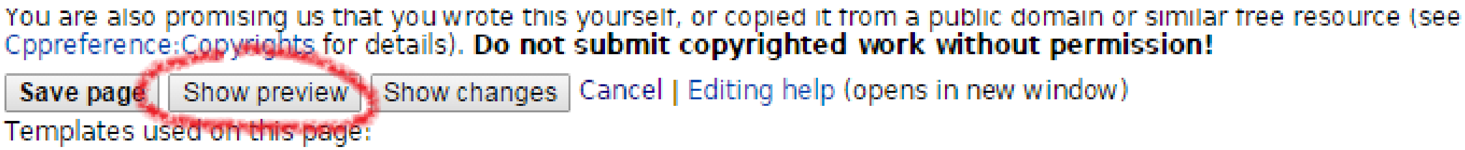
Return value

Inline templates



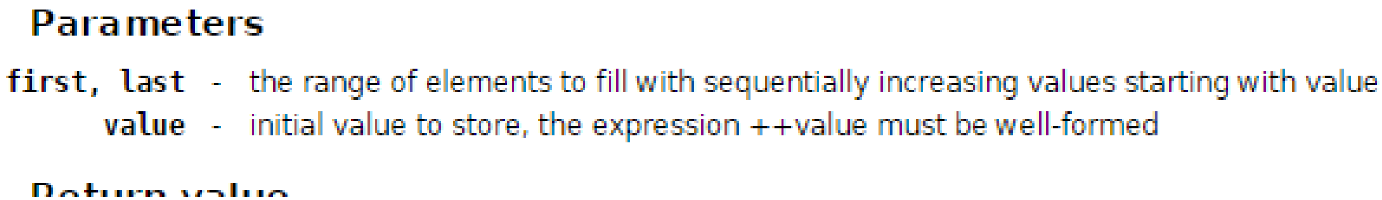
Editing cpp/algorithm/iota

```
===Parameters===
{{par begin}}
{{par | first, last | the range of elements to fill with sequentially increasing values starting with value }}
{{par | value | initial value to store, the expression ++value must be well-formed }}
{{par end}}
```



You are also promising us that you wrote this yourself, or copied it from a public domain or similar free resource (see Cppreference:Copyrights for details). **Do not submit copyrighted work without permission!**

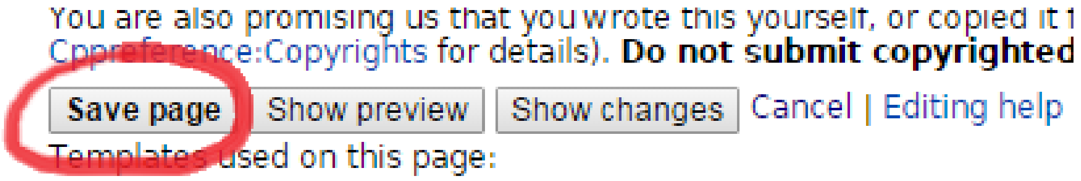
Templates used on this page:



Parameters

first, last - the range of elements to fill with sequentially increasing values starting with value

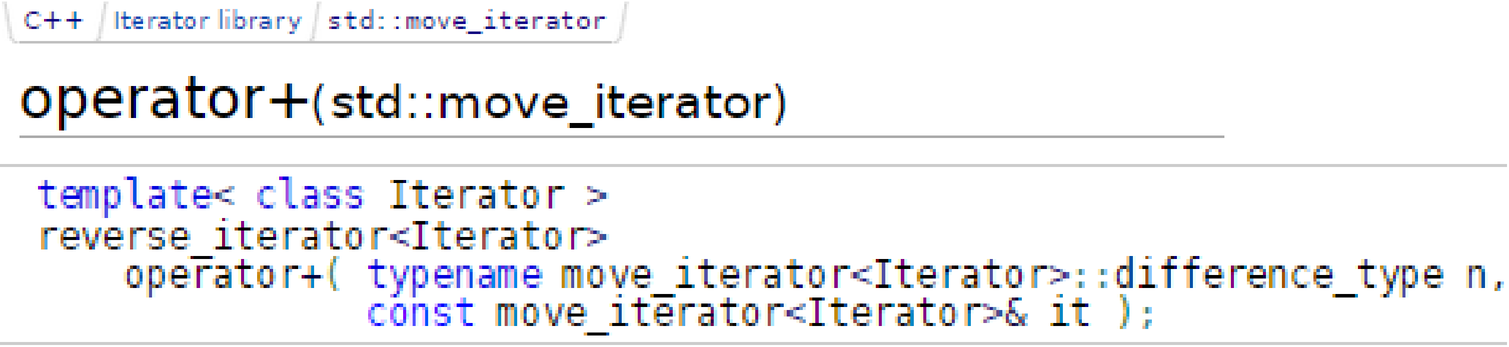
value - initial value to store, the expression ++value must be well-formed



You are also promising us that you wrote this yourself, or copied it f
Cppreference:Copyrights for details). **Do not submit copyrighted**

Templates used on this page:

“I clicked Edit and the box is empty”

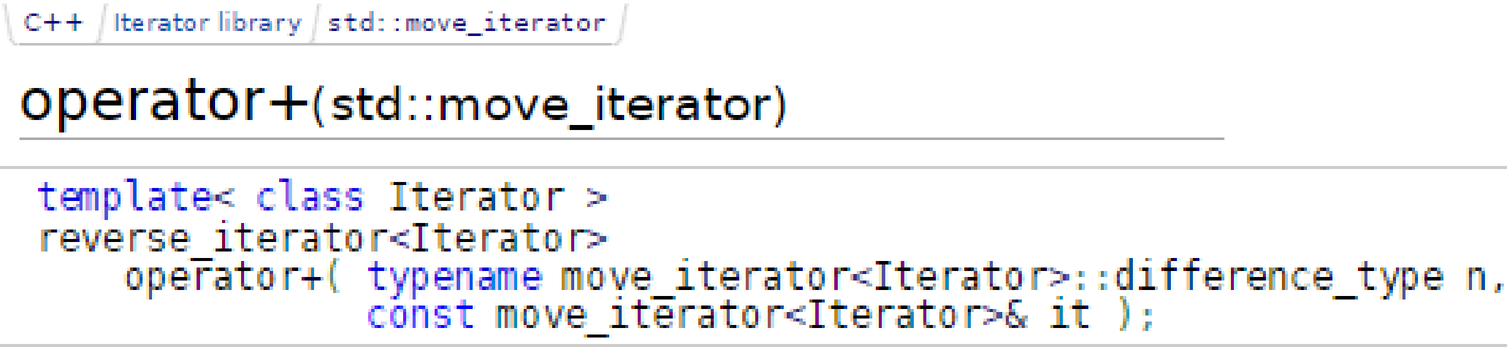


C++ / Iterator library / std::move_iterator

operator+(std::move_iterator)

```
template< class Iterator >
reverse_iterator<Iterator>
    operator+( typename move_iterator<Iterator>::difference_type n,
               const move_iterator<Iterator>& it );
```

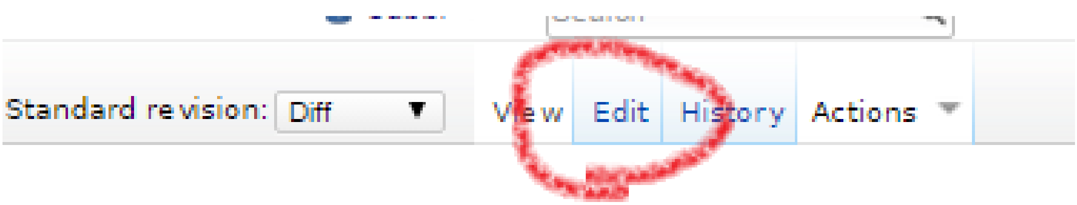
“I clicked Edit and the box is empty”



C++ / Iterator library / std::move_iterator

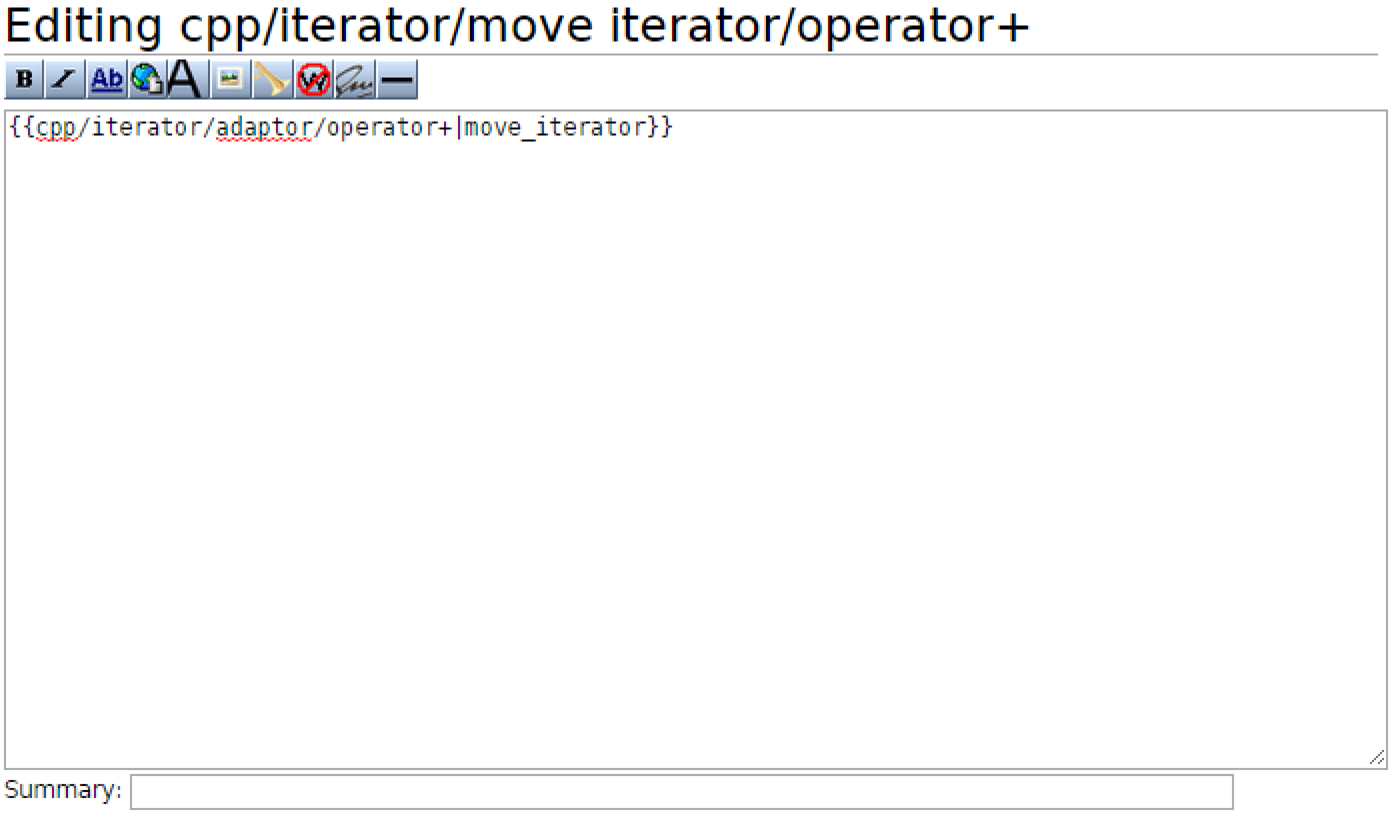
operator+(std::move_iterator)

```
template< class Iterator >
reverse_iterator<Iterator>
    operator+( typename move_iterator<Iterator>::difference_type n,
               const move_iterator<Iterator>& it );
```



Standard revision: Diff ▼

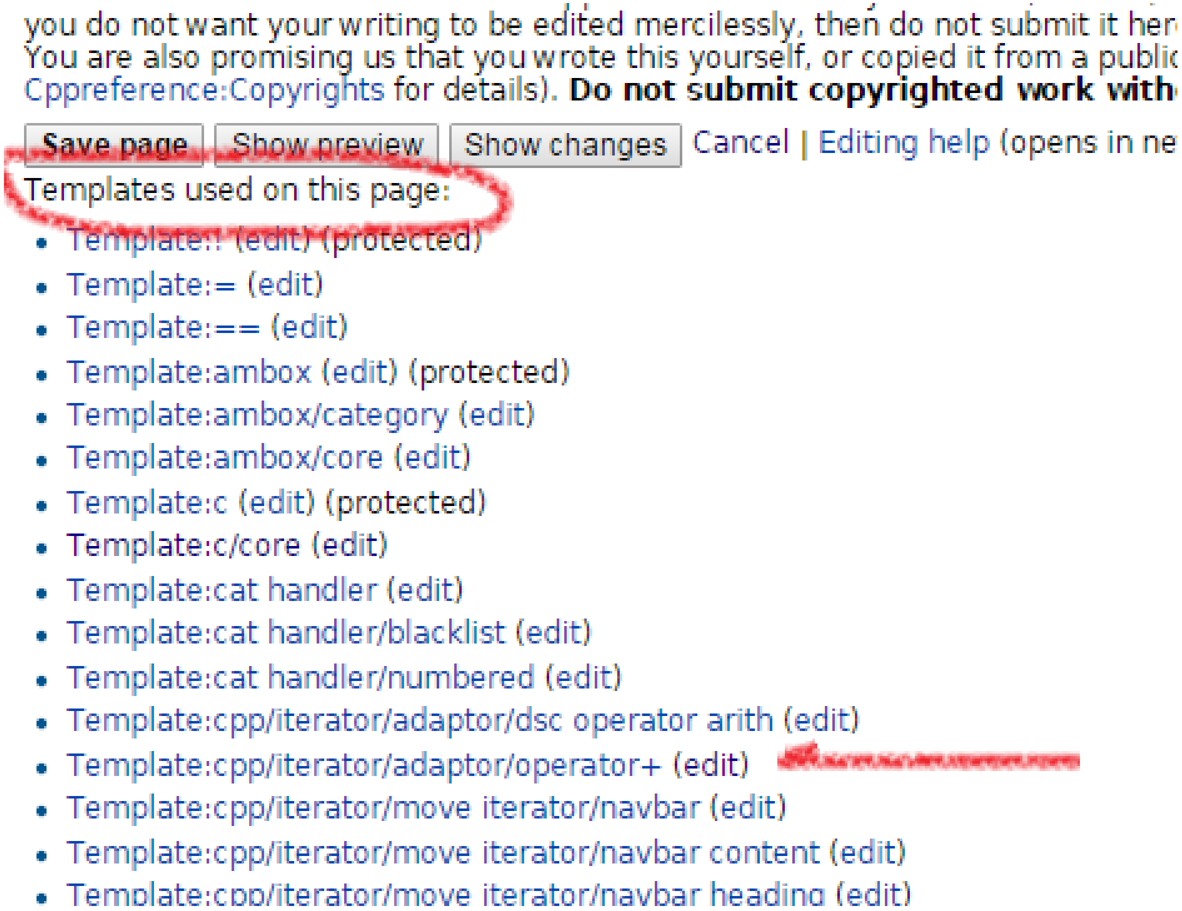
“I clicked Edit and the box is empty”



Editing cpp/iterator/move iterator/operator+

```
{{cpp/iterator/adaptor/operator+|move_iterator}}
```

“I clicked Edit and the box is empty”

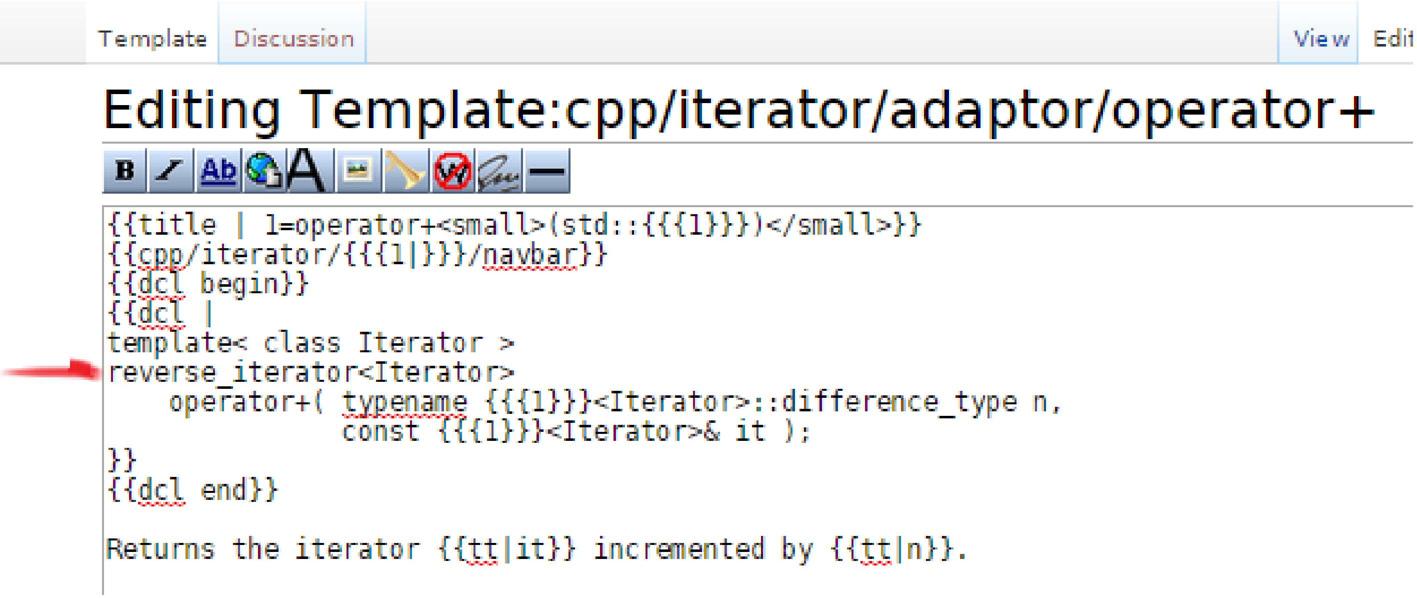


you do not want your writing to be edited mercilessly, then do not submit it her
You are also promising us that you wrote this yourself, or copied it from a public
Cppreference:Copyrights for details). **Do not submit copyrighted work with**

Templates used on this page:

- Template:! (edit) (protected)
- Template:= (edit)
- Template:== (edit)
- Template:ambox (edit) (protected)
- Template:ambox/category (edit)
- Template:ambox/core (edit)
- Template:c (edit) (protected)
- Template:c/core (edit)
- Template:cat handler (edit)
- Template:cat handler/blacklist (edit)
- Template:cat handler/numbered (edit)
- Template:cpp/iterator/adaptor/dsc operator arith (edit)
- Template:cpp/iterator/adaptor/operator+ (edit)
- Template:cpp/iterator/move iterator/navbar (edit)
- Template:cpp/iterator/move iterator/navbar content (edit)
- Template:cpp/iterator/move iterator/navbar heading (edit)

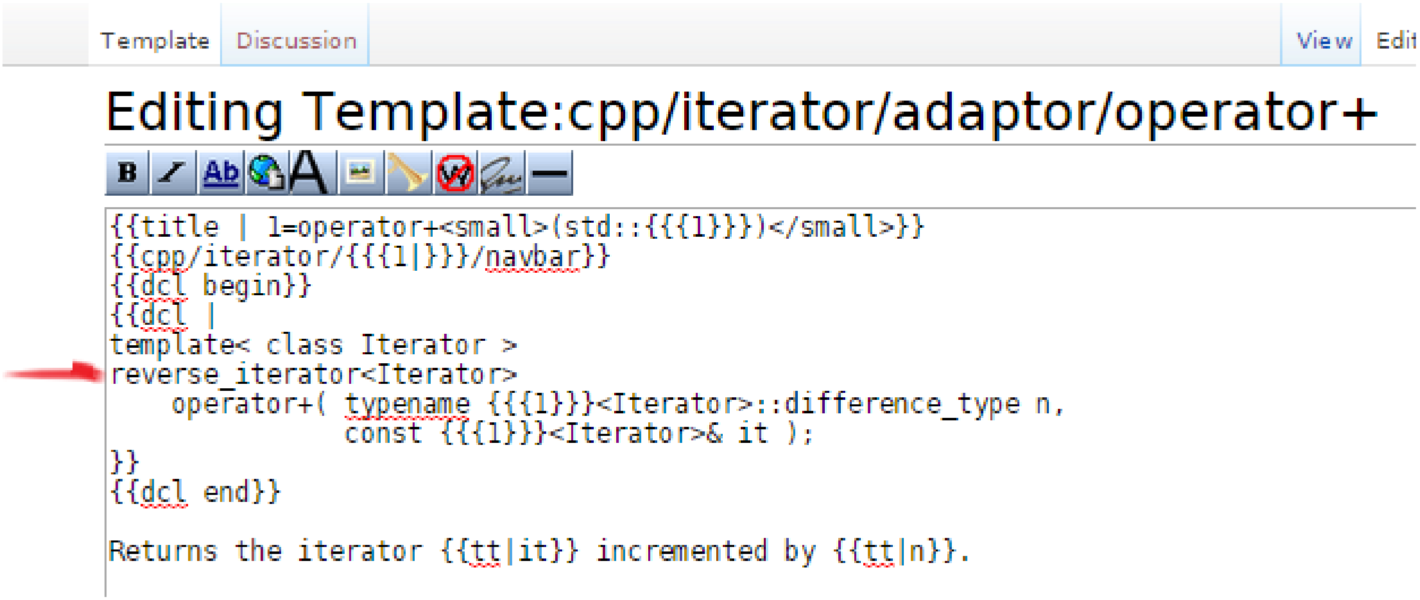
“I clicked Edit and the box is empty”



Editing Template:cpp/iterator/adaptor/operator+

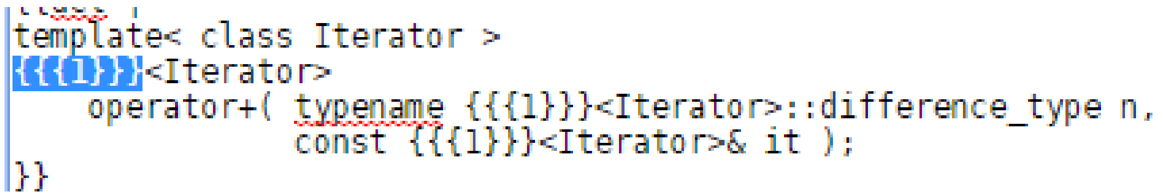
```
{{title | 1=operator+<small>(std::{{{1}}})</small>}}
{{cpp/iterator/{{{1|}}}/navbar}}
{{dcl begin}}
{{dcl |
template< class Iterator >
reverse_iterator<Iterator>
    operator+( typename {{{1}}}<Iterator>::difference_type n,
               const {{{1}}}<Iterator>& it );
}}
{{dcl end}}

Returns the iterator {{tt|it}} incremented by {{tt|n}}.
```

“I clicked Edit and the box is empty”



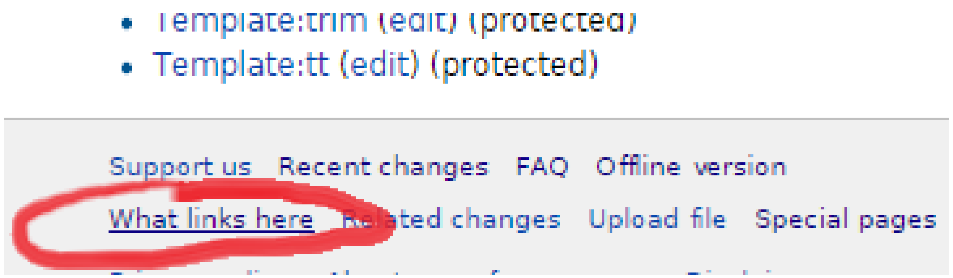
Editing Template:cpp/iterator/adaptor/operator+

```
{{title | 1=operator+<small>(std::{{{1}}})</small>}}
{{cpp/iterator/{{{1|}}}/navbar}}
{{dcl begin}}
{{dcl |
template< class Iterator >
reverse_iterator<Iterator>
    operator+( typename {{{1}}}<Iterator>::difference_type n,
               const {{{1}}}<Iterator>& it );
}}
{{dcl end}}

Returns the iterator {{tt|it}} incremented by {{tt|n}}.
```



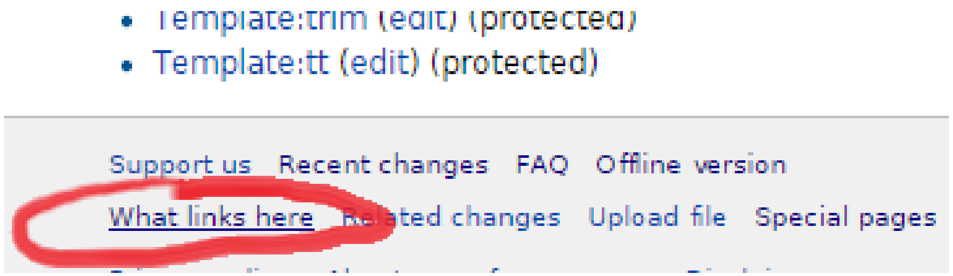
```
template< class Iterator >
{{{1}}}<Iterator>
    operator+( typename {{{1}}}<Iterator>::difference_type n,
               const {{{1}}}<Iterator>& it );
}}
```

“I clicked Edit and the box is empty”

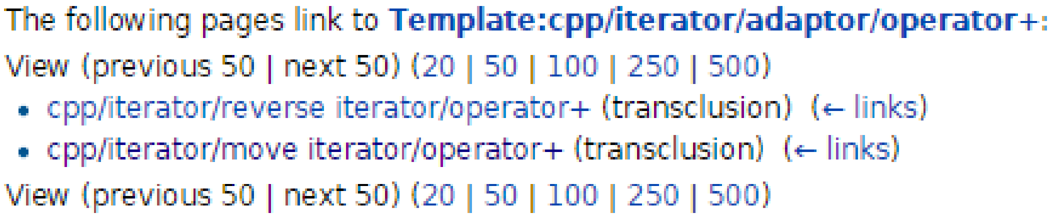


- Template:trim (edit) (protected)
- Template:tt (edit) (protected)

“I clicked Edit and the box is empty”



- Template:trim (edit) (protected)
- Template:tt (edit) (protected)



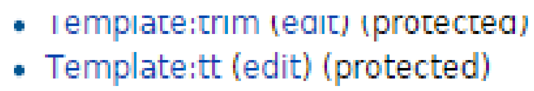
The following pages link to **Template:cpp/iterator/adaptor/operator+**:

View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)

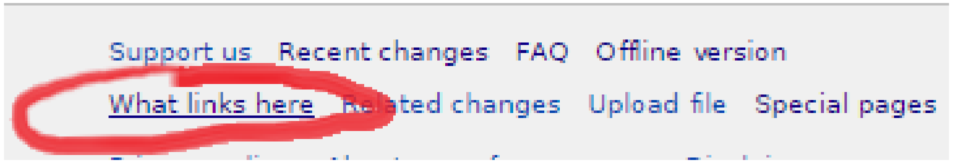
- cpp/iterator/reverse iterator/operator+ (transclusion) (← links)
- cpp/iterator/move iterator/operator+ (transclusion) (← links)

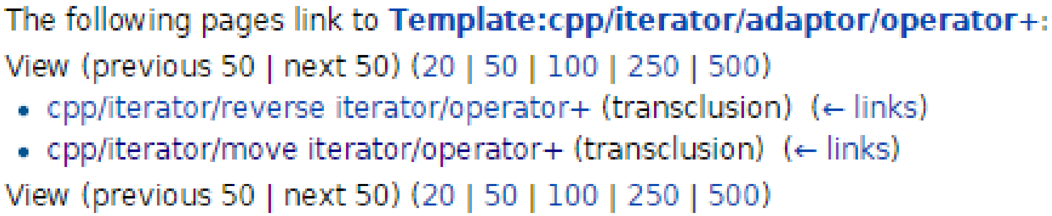
View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)

“I clicked Edit and the box is empty”



- Template:trim (edit) (protected)
- Template:tt (edit) (protected)



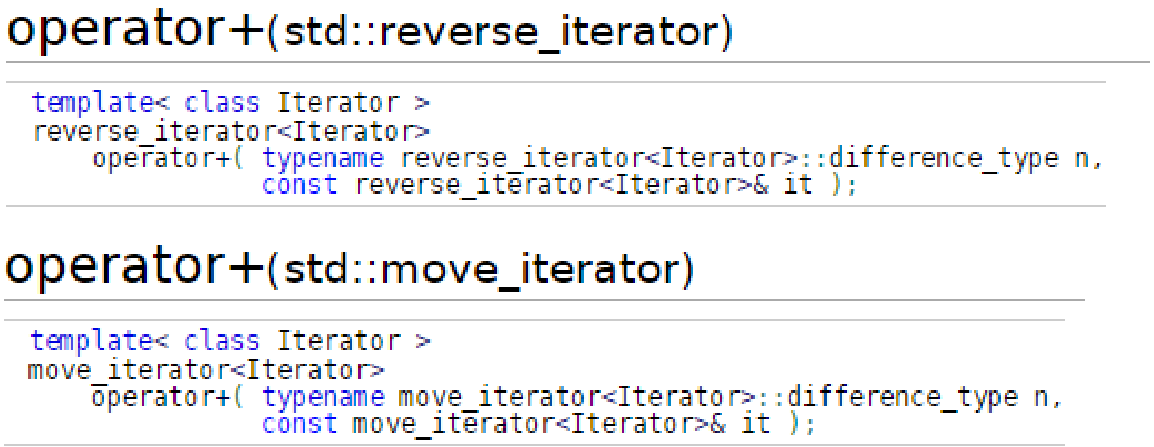


The following pages link to **Template:cpp/iterator/adaptor/operator+**:

View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)

- cpp/iterator/reverse iterator/operator+ (transclusion) (← links)
- cpp/iterator/move iterator/operator+ (transclusion) (← links)

View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)



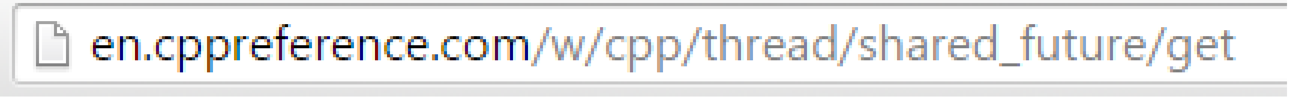
operator+(std::reverse_iterator)

```
template< class Iterator >
reverse_iterator<Iterator>
    operator+( typename reverse_iterator<Iterator>::difference_type n,
               const reverse_iterator<Iterator>& it );
```

operator+(std::move_iterator)

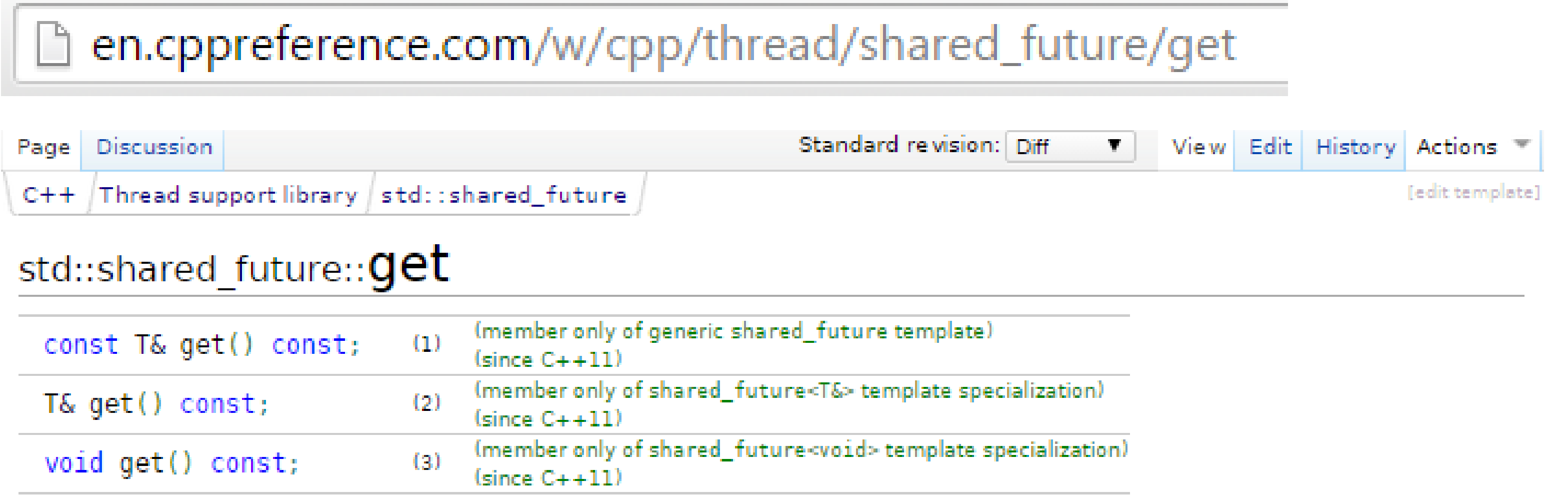
```
template< class Iterator >
move_iterator<Iterator>
    operator+( typename move_iterator<Iterator>::difference_type n,
               const move_iterator<Iterator>& it );
```

Conditionals



en.cppreference.com/w/cpp/thread/shared_future/get

Conditionals

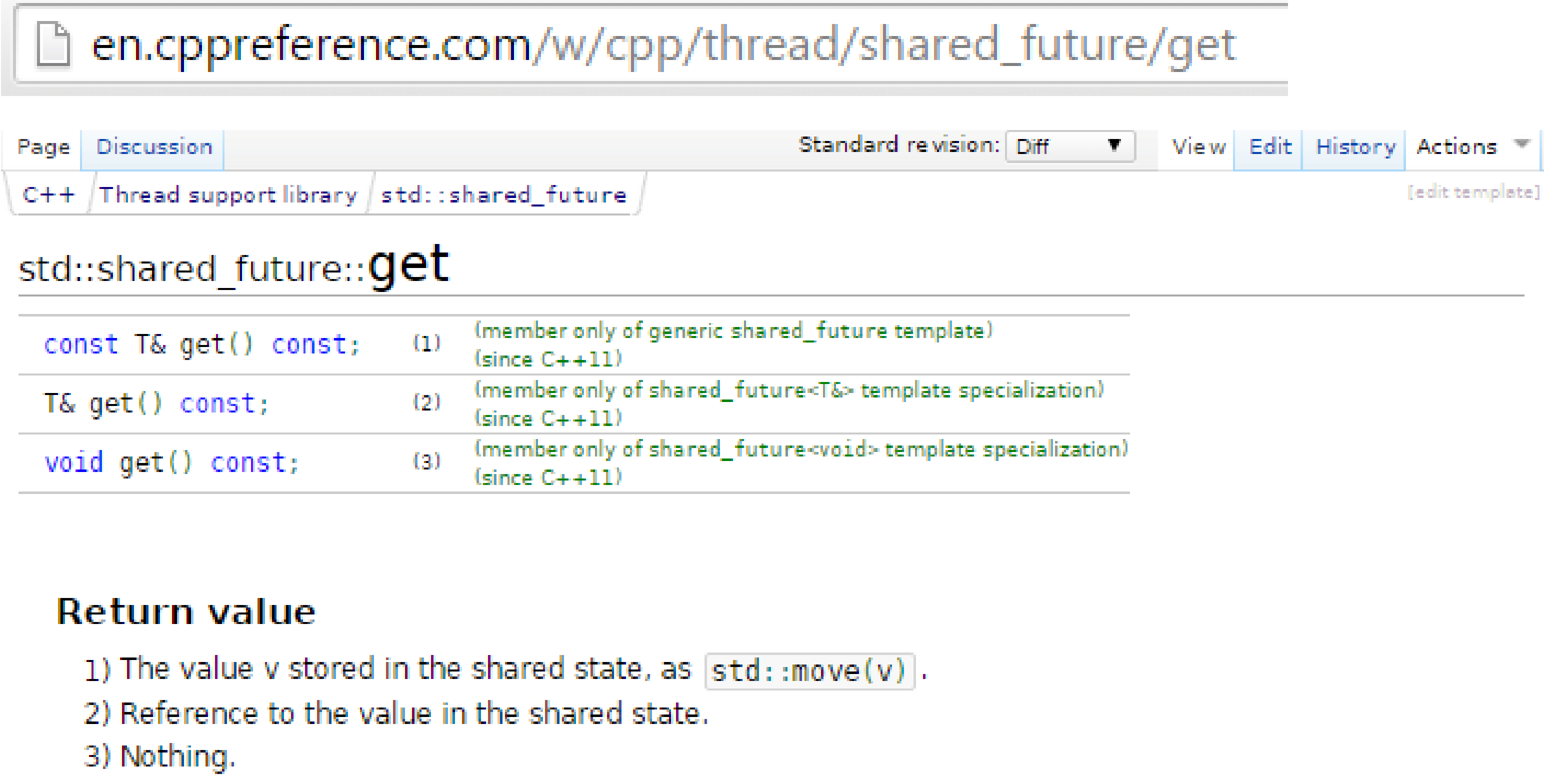


en.cppreference.com/w/cpp/thread/shared_future/get

std::shared_future::get

`const T& get() const;`	(1)	(member only of generic shared_future template) (since C++11)
`T& get() const;`	(2)	(member only of shared_future<T&> template specialization) (since C++11)
`void get() const;`	(3)	(member only of shared_future<void> template specialization) (since C++11)

Conditionals



std::shared_future::get

`const T& get() const;`	(1)	(member only of generic shared_future template) (since C++11)
`T& get() const;`	(2)	(member only of shared_future<T&> template specialization) (since C++11)
`void get() const;`	(3)	(member only of shared_future<void> template specialization) (since C++11)

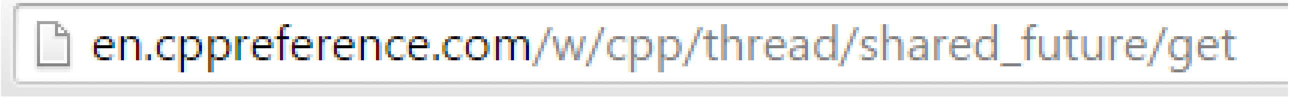
Return value

1) The value v stored in the shared state, as `std::move(v)`.

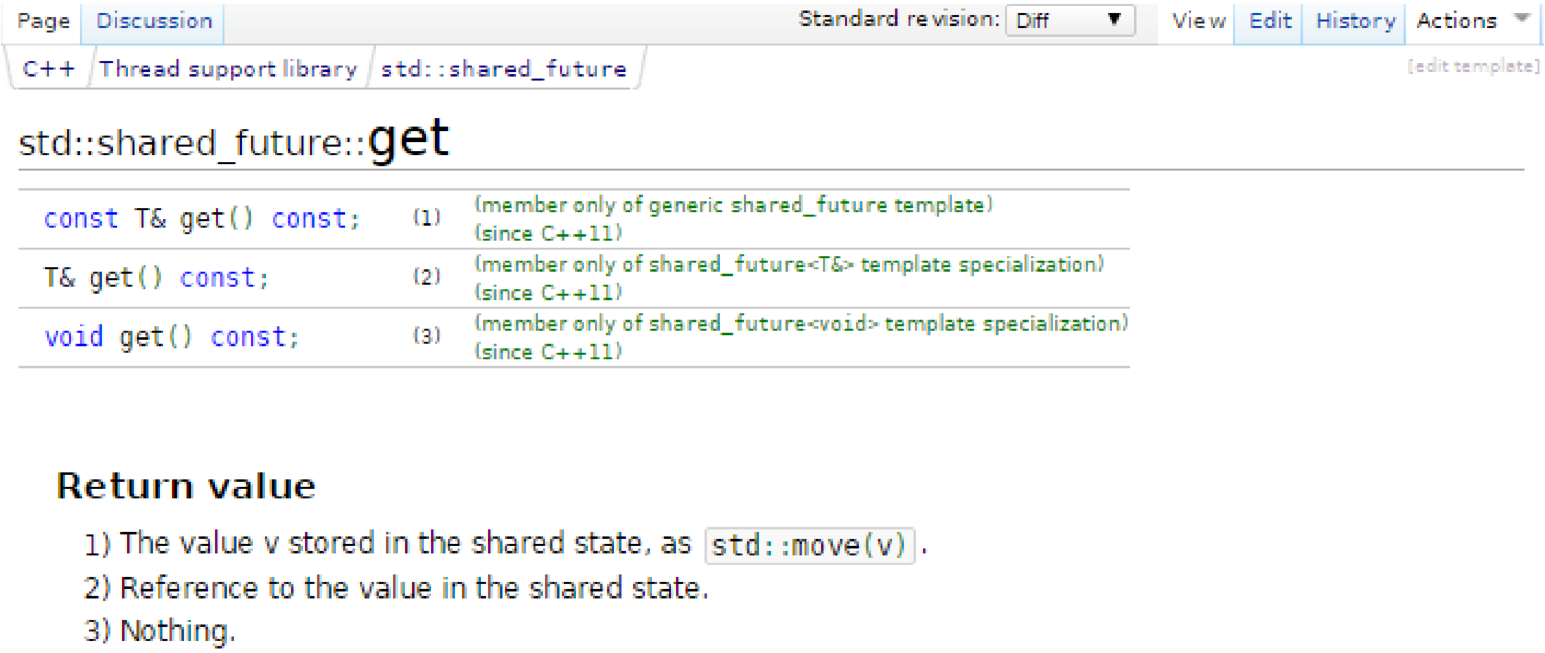
2) Reference to the value in the shared state.

3) Nothing.

Conditionals



en.cppreference.com/w/cpp/thread/shared_future/get



std::shared_future::get

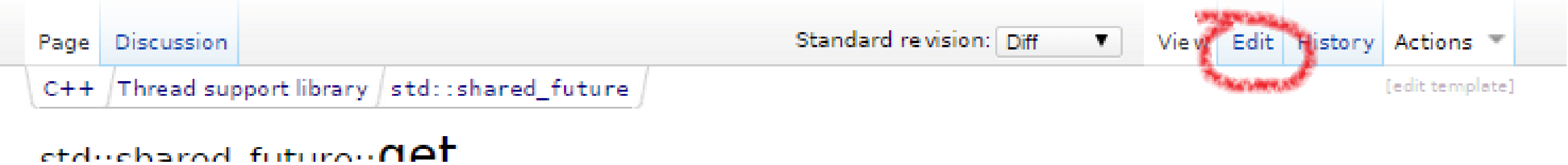
`const T& get() const;`	(1)	(member only of generic shared_future template) (since C++11)
`T& get() const;`	(2)	(member only of shared_future<T&> template specialization) (since C++11)
`void get() const;`	(3)	(member only of shared_future<void> template specialization) (since C++11)

Return value

1) The value v stored in the shared state, as `std::move(v)`.

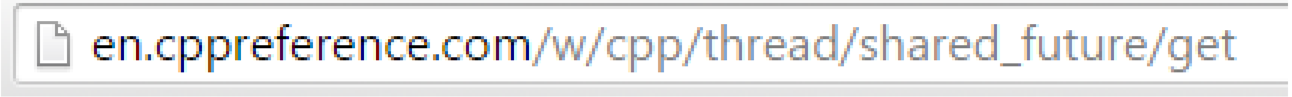
2) Reference to the value in the shared state.

3) Nothing.

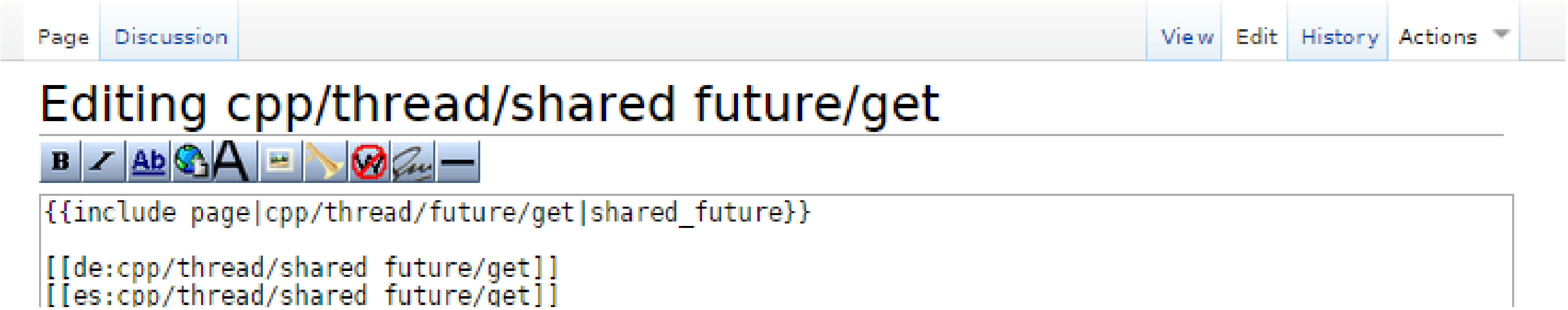


std::shared_future::get

Conditionals

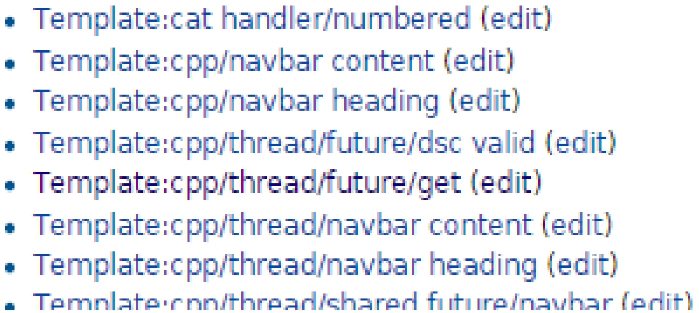


en.cppreference.com/w/cpp/thread/shared_future/get



Editing cpp/thread/shared future/get

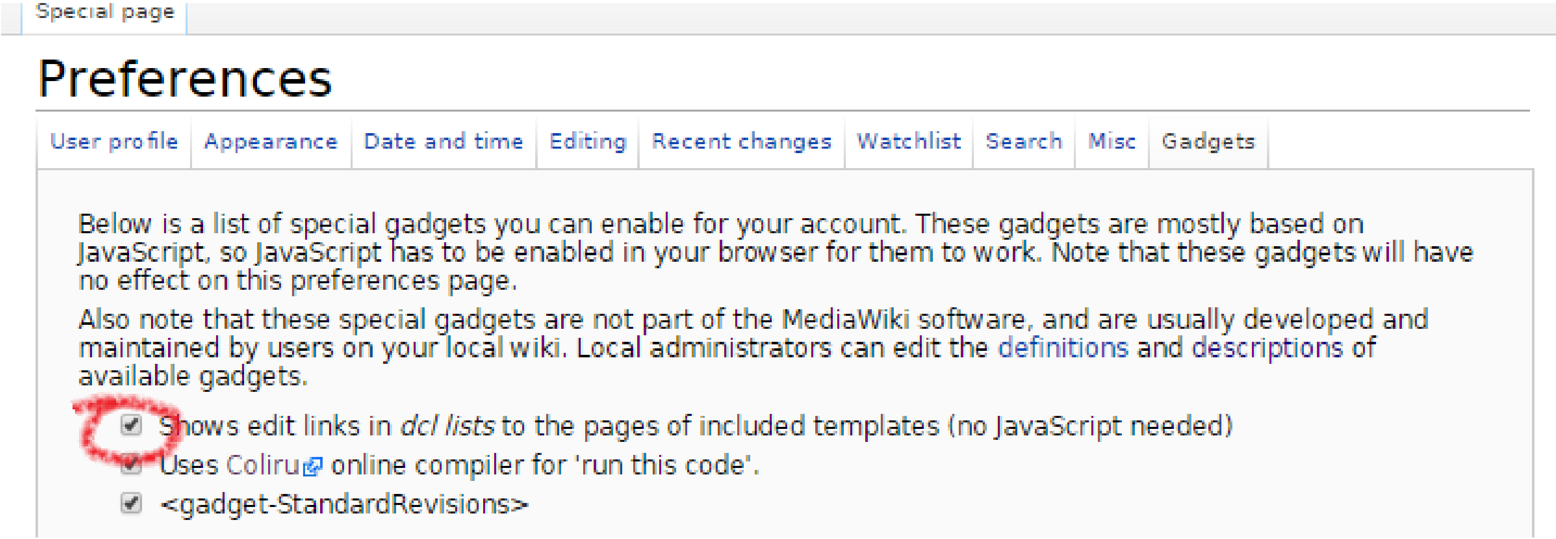
```
{{include page|cpp/thread/future/get|shared_future}}

[[de:cpp/thread/shared future/get]]
[[es:cpp/thread/shared future/get]]
```



- Template:cat handler/numbered (edit)
- Template:cpp/navbar content (edit)
- Template:cpp/navbar heading (edit)
- Template:cpp/thread/future/dsc valid (edit)
- Template:cpp/thread/future/get (edit)
- Template:cpp/thread/navbar content (edit)
- Template:cpp/thread/navbar heading (edit)
- Template:cpp/thread/shared future/navbar (edit)

Conditionals

Or, if you do that a lot…



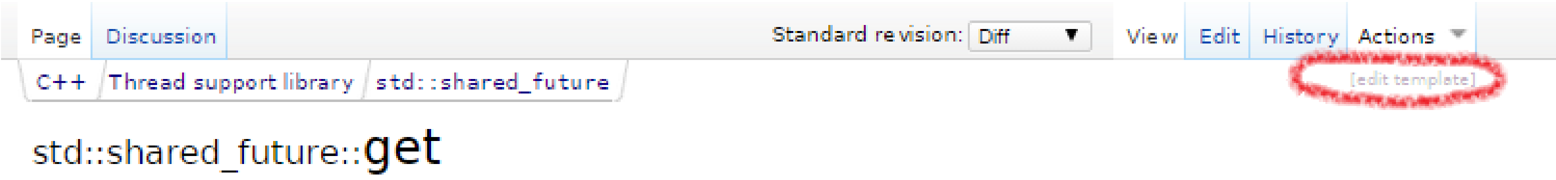
Special page

Preferences

Below is a list of special gadgets you can enable for your account. These gadgets are mostly based on JavaScript, so JavaScript has to be enabled in your browser for them to work. Note that these gadgets will have no effect on this preferences page.

Also note that these special gadgets are not part of the MediaWiki software, and are usually developed and maintained by users on your local wiki. Local administrators can edit the definitions and descriptions of available gadgets.

- ☑ Shows edit links in *dcl lists* to the pages of included templates (no JavaScript needed)
- ☑ Uses Coliru online compiler for 'run this code'.
- ☑ <gadget-StandardRevisions>

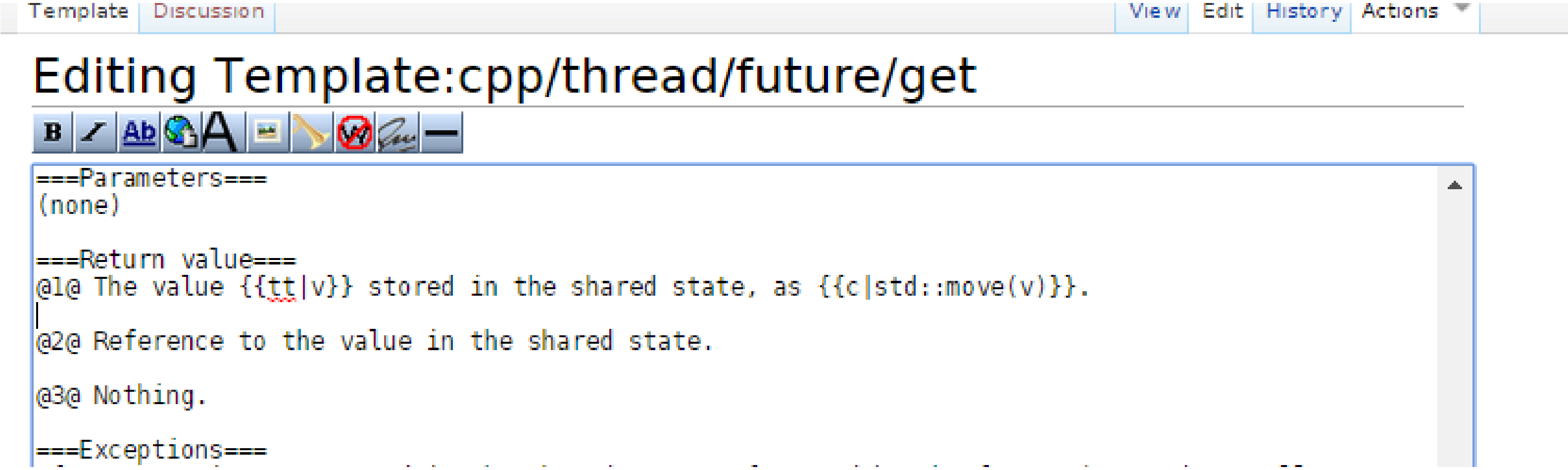


C++ / Thread support library / std::shared_future

[edit template]

std::shared_future::get

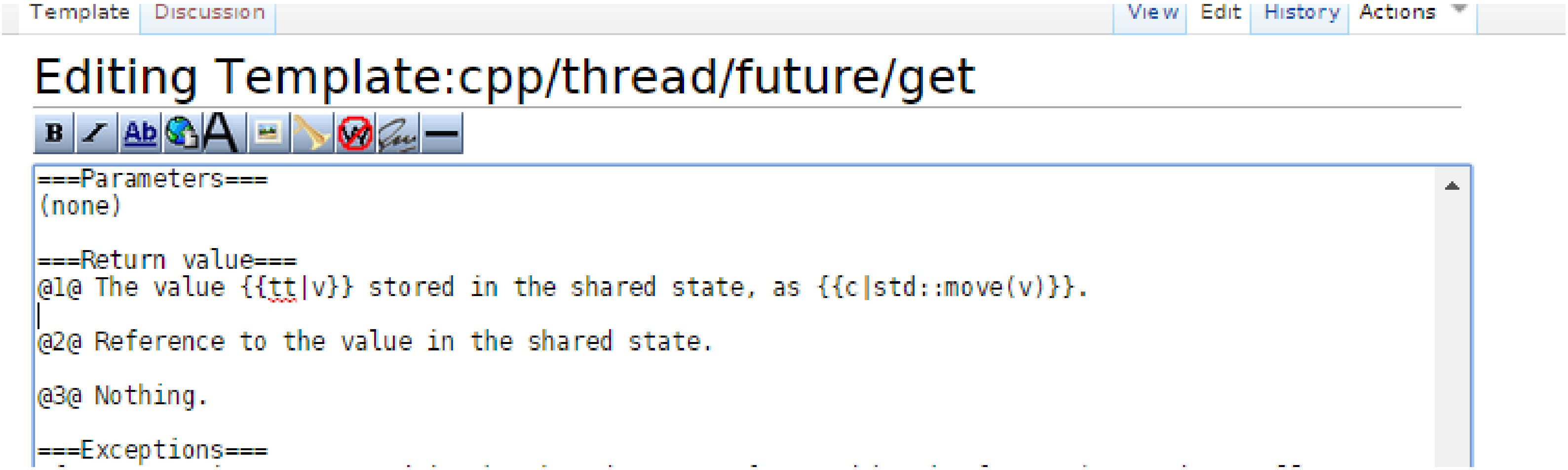
Conditionals



Editing Template:cpp/thread/future/get

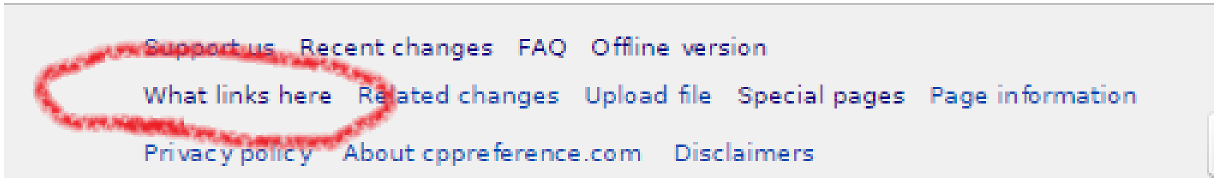
```
===Parameters===
(none)

===Return value===
@1@ The value {{tt|v}} stored in the shared state, as {{c|std::move(v)}}.

@2@ Reference to the value in the shared state.

@3@ Nothing.

===Exceptions===
```

Conditionals

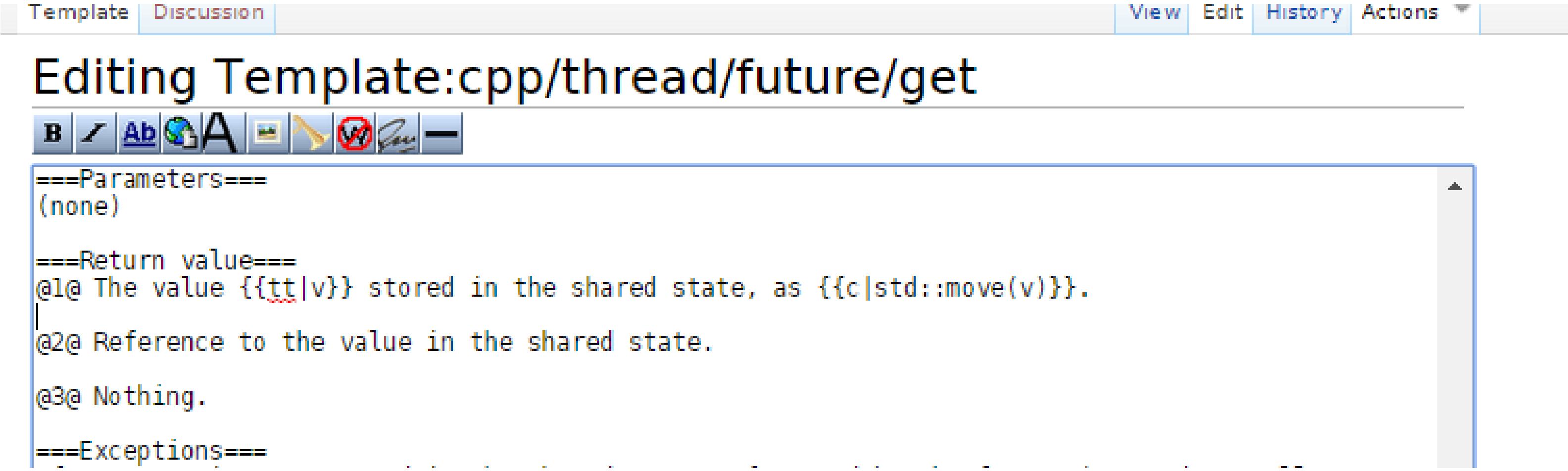


Editing Template:cpp/thread/future/get

```
===Parameters===
(none)

===Return value===
@1@ The value {{tt|v}} stored in the shared state, as {{c|std::move(v)}}.

@2@ Reference to the value in the shared state.

@3@ Nothing.

===Exceptions===
```

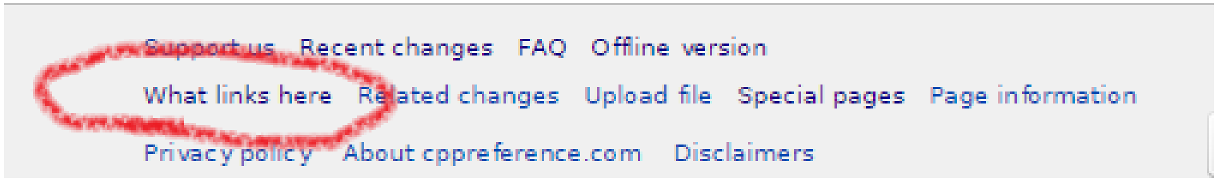


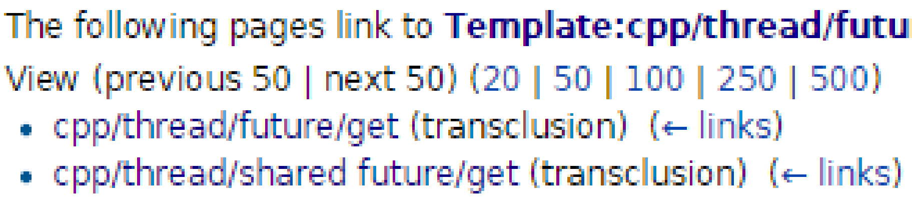
Conditionals



Editing Template:cpp/thread/future/get

```
===Parameters===
(none)

===Return value===
@1@ The value {{tt|v}} stored in the shared state, as {{c|std::move(v)}}.

@2@ Reference to the value in the shared state.

@3@ Nothing.

===Exceptions===
```

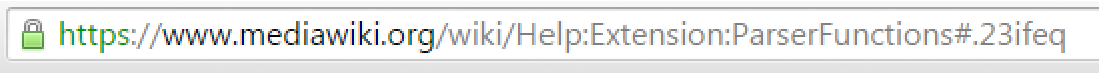



The following pages link to **Template:cpp/thread/future/get**:

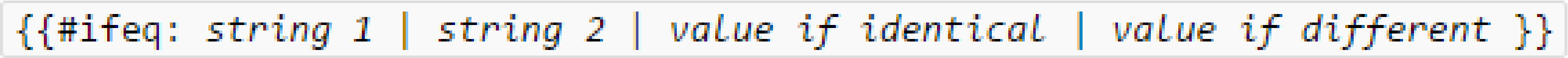
View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)

- cpp/thread/future/get (transclusion) (← links)
- cpp/thread/shared future/get (transclusion) (← links)

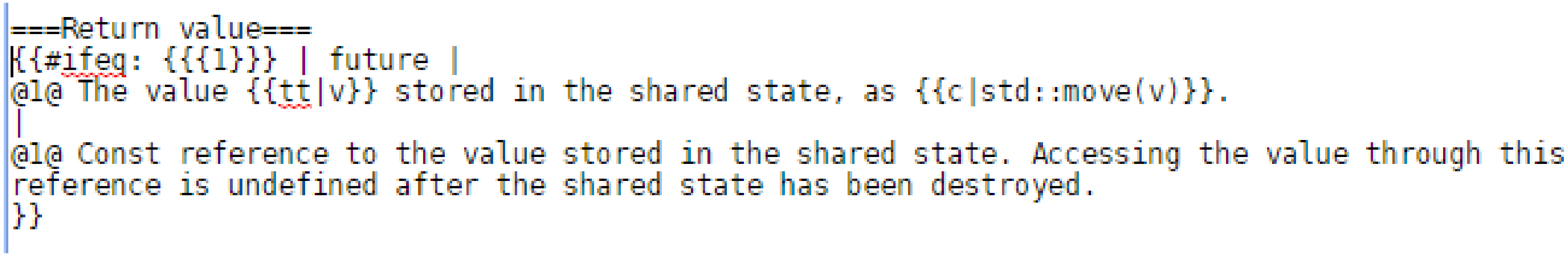
Conditionals



https://www.mediawiki.org/wiki/Help:Extension:ParserFunctions#.23ifeq

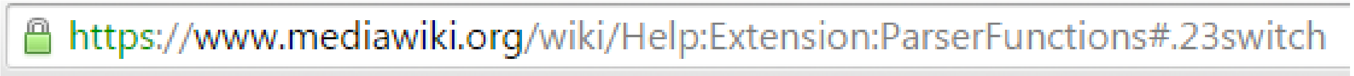


```
{{#ifeq: string 1 | string 2 | value if identical | value if different }}
```

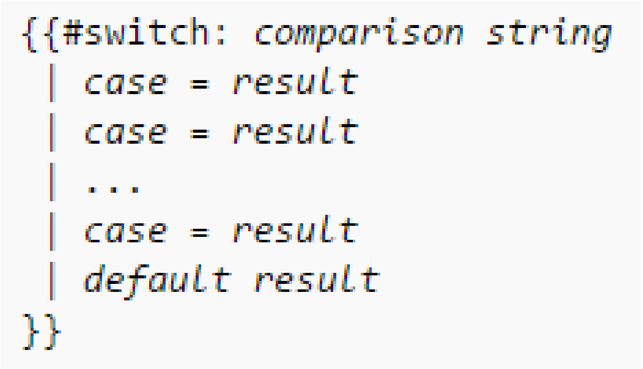


```
===Return value===
{{#ifeq: {{{1}}} | future |
@1@ The value {{tt|v}} stored in the shared state, as {{c|std::move(v)}}.
|
@1@ Const reference to the value stored in the shared state. Accessing the value through this
reference is undefined after the shared state has been destroyed.
}}
```

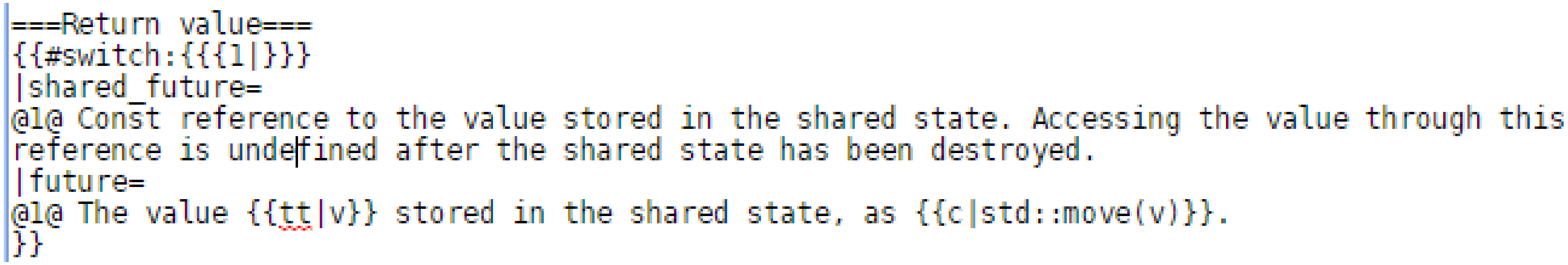
Conditionals



https://www.mediawiki.org/wiki/Help:Extension:ParserFunctions#.23switch

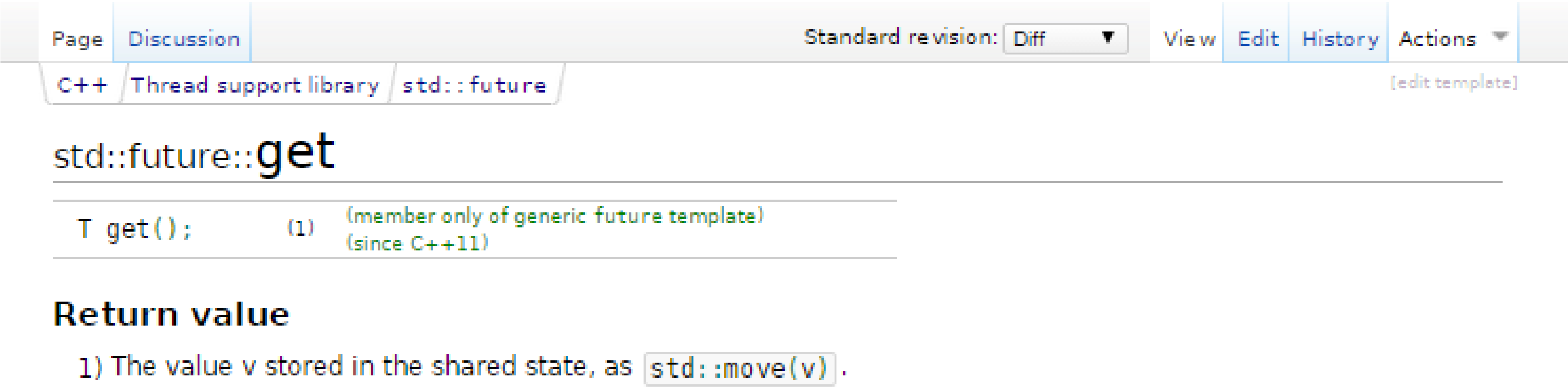


```
{{#switch: comparison string
 | case = result
 | case = result
 | ...
 | case = result
 | default result
}}
```



```
===Return value===
{{#switch:{{{1|}}}
|shared_future=
@l@ Const reference to the value stored in the shared state. Accessing the value through this
reference is undefined after the shared state has been destroyed.
|future=
@l@ The value {{tt|v}} stored in the shared state, as {{c|std::move(v)}}.
}}
```

Conditionals

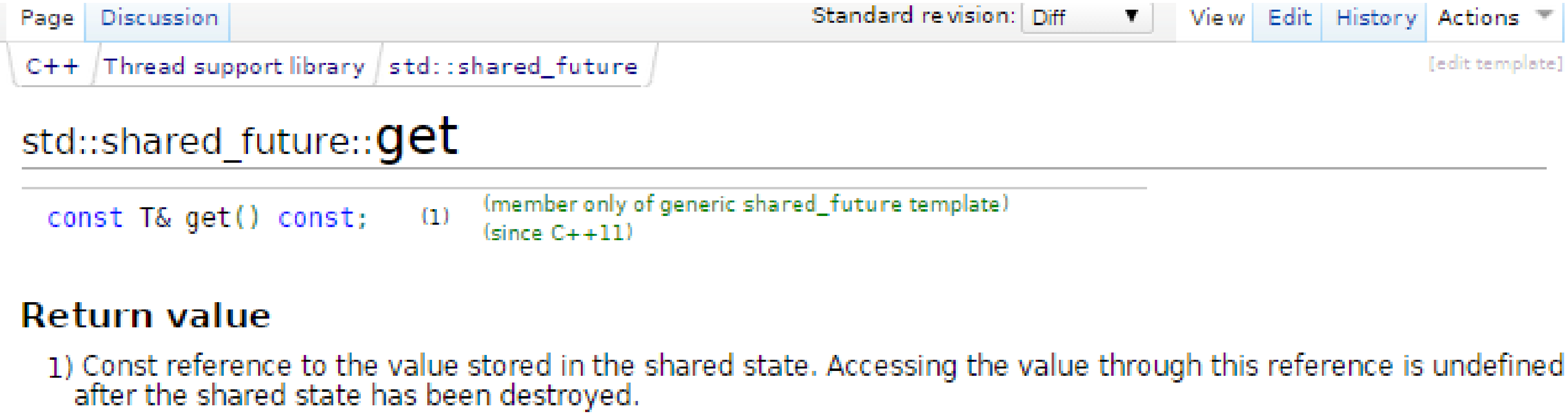


std::future::get

`T get();`	(1)	(member only of generic future template) (since C++11)

Return value

1) The value v stored in the shared state, as `std::move(v)`.



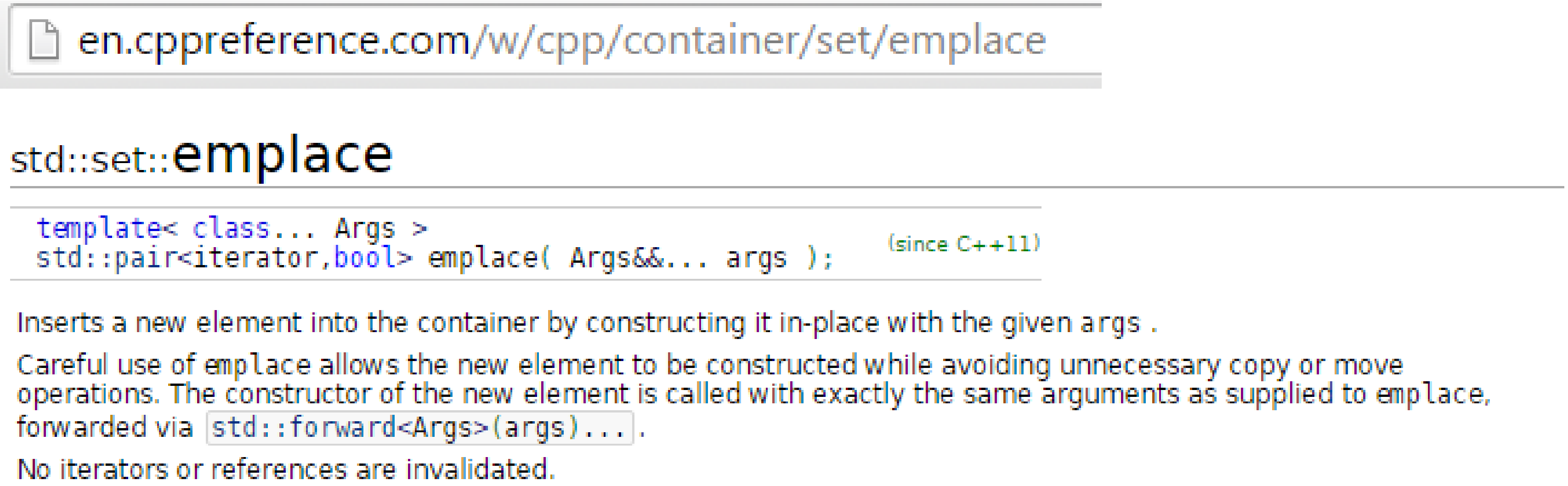
std::shared_future::get

`const T& get() const;`	(1)	(member only of generic shared_future template) (since C++11)

Return value

1) Const reference to the value stored in the shared state. Accessing the value through this reference is undefined after the shared state has been destroyed.

C++ specific conditionals



en.cppreference.com/w/cpp/container/set/emplace

std::set::emplace

```
template< class... Args >
std::pair<iterator,bool> emplace( Args&&... args );
```

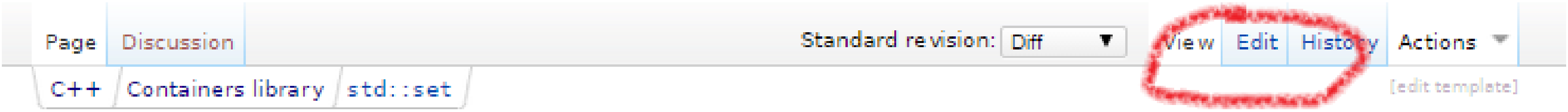
(since C++11)

Inserts a new element into the container by constructing it in-place with the given `args` .

Careful use of `emplace` allows the new element to be constructed while avoiding unnecessary copy or move operations. The constructor of the new element is called with exactly the same arguments as supplied to `emplace`, forwarded via `std::forward<Args>(args)...` .

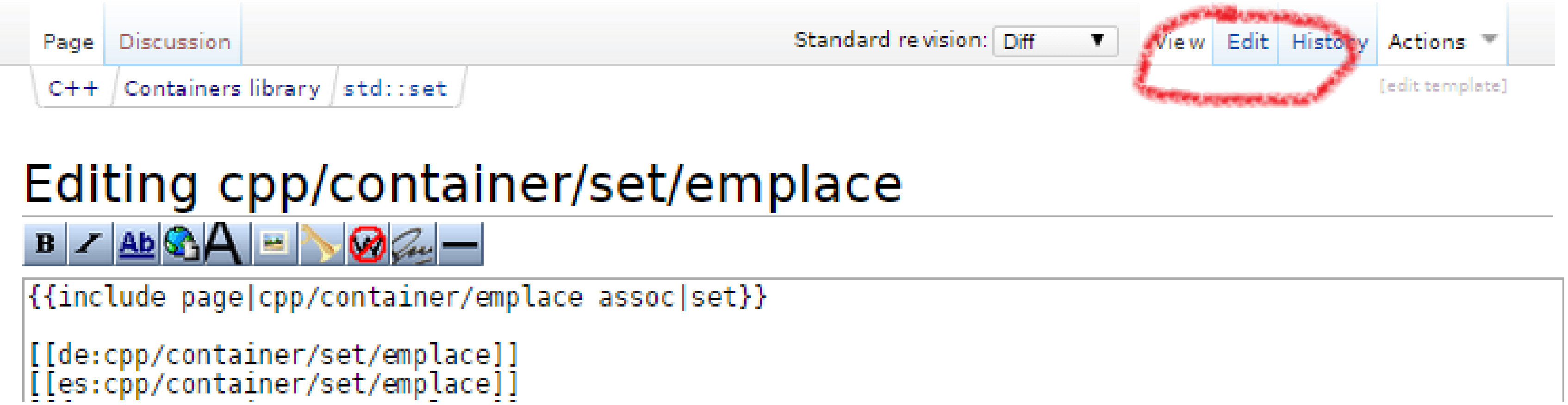
No iterators or references are invalidated.

C++ specific conditionals



Standard revision: Diff

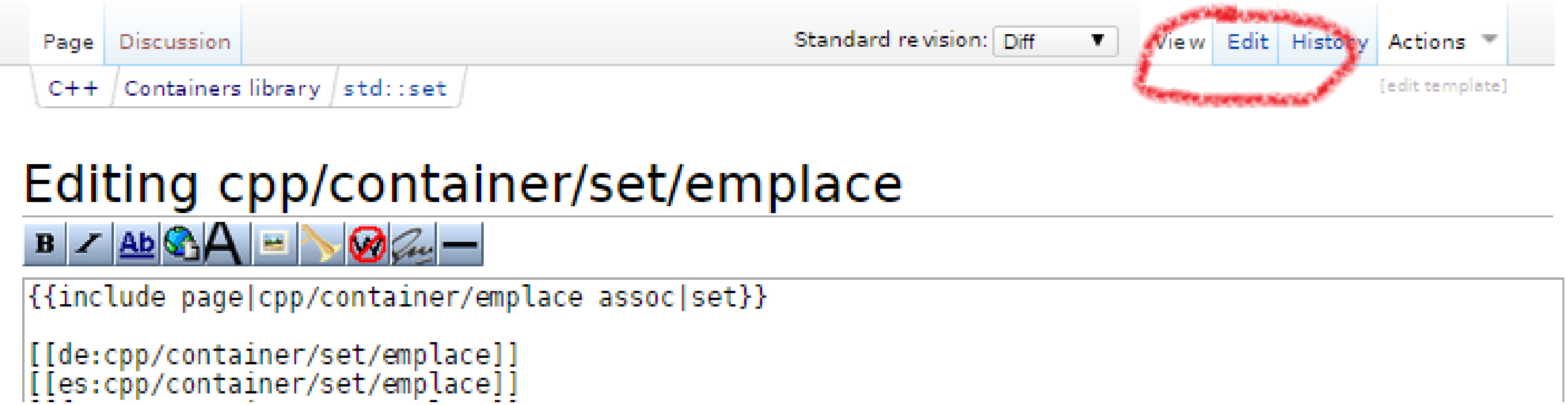
C++ specific conditionals



Editing cpp/container/set/emplace

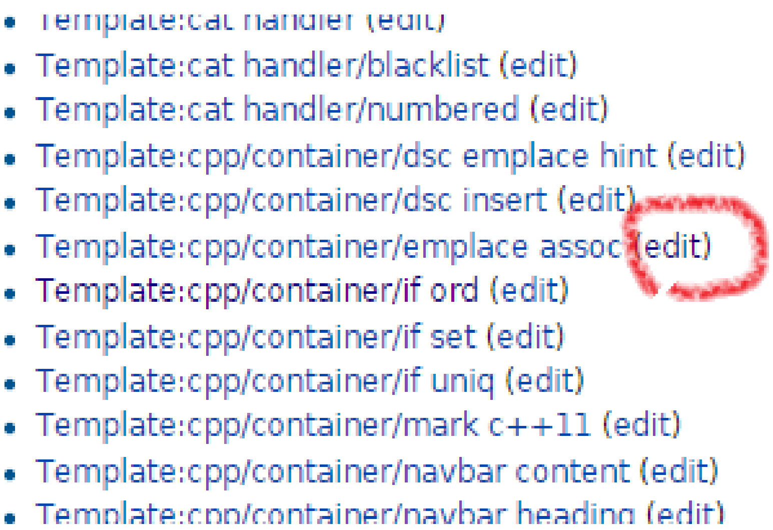
```
{{include page|cpp/container/emplace assoc|set}}

[[de:cpp/container/set/emplace]]
[[es:cpp/container/set/emplace]]
```

C++ specific conditionals



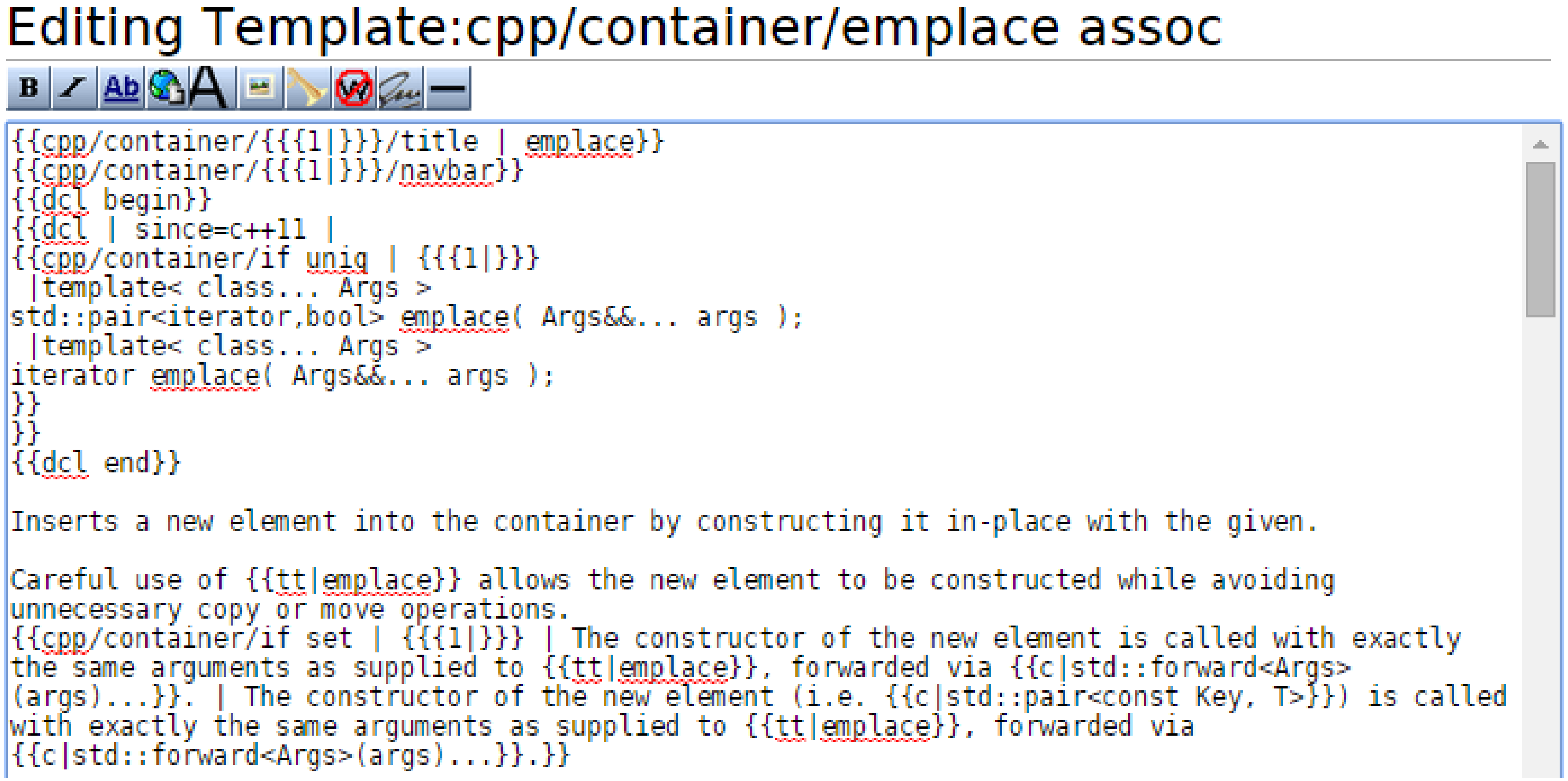
Editing cpp/container/set/emplace

```
{{include page|cpp/container/emplace assoc|set}}

[[de:cpp/container/set/emplace]]
[[es:cpp/container/set/emplace]]
```



- Template:cat handler (edit)
- Template:cat handler/blacklist (edit)
- Template:cat handler/numbered (edit)
- Template:cpp/container/dsc emplace hint (edit)
- Template:cpp/container/dsc insert (edit)
- Template:cpp/container/emplace assoc (edit)
- Template:cpp/container/if ord (edit)
- Template:cpp/container/if set (edit)
- Template:cpp/container/if uniq (edit)
- Template:cpp/container/mark c++11 (edit)
- Template:cpp/container/navbar content (edit)
- Template:cpp/container/navbar heading (edit)

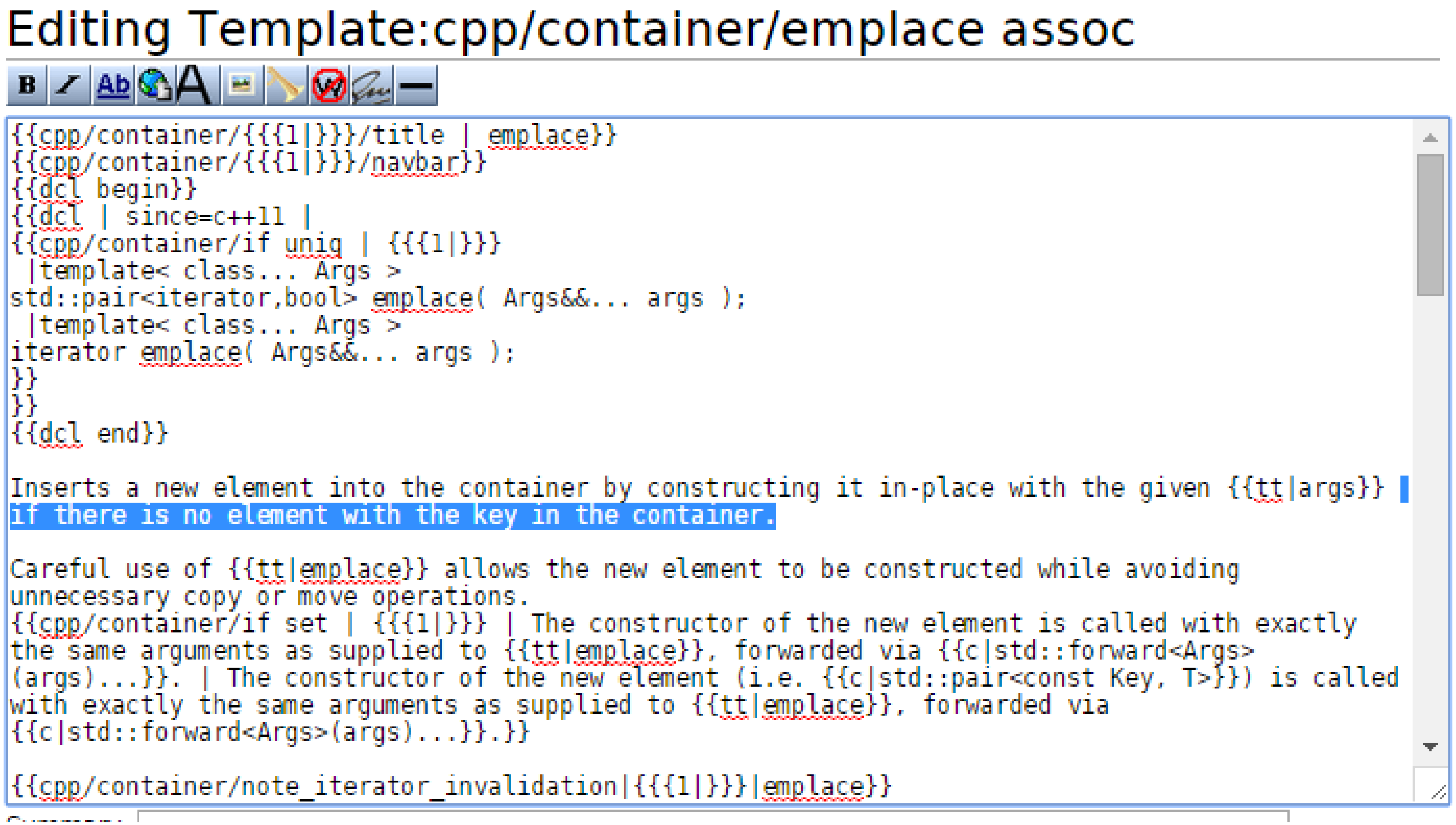
C++ specific conditionals



Editing Template:cpp/container/emplace assoc

```
{{cpp/container/{{{1|}}}/title | emplace}}
{{cpp/container/{{{1|}}}/navbar}}
{{dcl begin}}
{{dcl | since=c++11 |
{{cpp/container/if uniq | {{{1|}}}
 |template< class... Args >
std::pair<iterator,bool> emplace( Args&&... args );
 |template< class... Args >
iterator emplace( Args&&... args );
}}
}}
{{dcl end}}

Inserts a new element into the container by constructing it in-place with the given.

Careful use of {{tt|emplace}} allows the new element to be constructed while avoiding
unnecessary copy or move operations.
{{cpp/container/if set | {{{1|}}} | The constructor of the new element is called with exactly
the same arguments as supplied to {{tt|emplace}}, forwarded via {{c|std::forward<Args>
(args)...}}. | The constructor of the new element (i.e. {{c|std::pair<const Key, T>}}) is called
with exactly the same arguments as supplied to {{tt|emplace}}, forwarded via
{{c|std::forward<Args>(args)...}}.}}
```

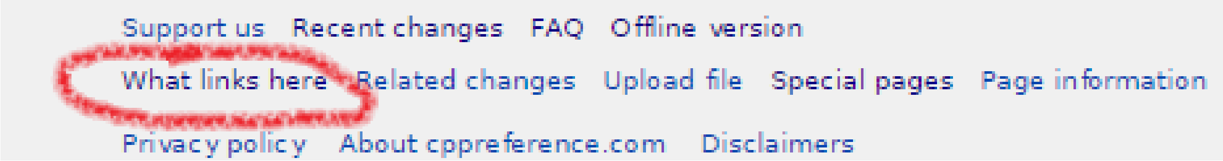
C++ specific conditionals



Editing Template:cpp/container/emplace assoc

```
{{cpp/container/{{{1|}}}/title | emplace}}
{{cpp/container/{{{1|}}}/navbar}}
{{dcl begin}}
{{dcl | since=c++11 |
{{cpp/container/if uniq | {{{1|}}}
 |template< class... Args >
std::pair<iterator,bool> emplace( Args&&... args );
 |template< class... Args >
iterator emplace( Args&&... args );
}}
}}
{{dcl end}}

Inserts a new element into the container by constructing it in-place with the given {{tt|args}}
if there is no element with the key in the container.

Careful use of {{tt|emplace}} allows the new element to be constructed while avoiding
unnecessary copy or move operations.
{{cpp/container/if set | {{{1|}}} | The constructor of the new element is called with exactly
the same arguments as supplied to {{tt|emplace}}, forwarded via {{c|std::forward<Args>
(args)...}}. | The constructor of the new element (i.e. {{c|std::pair<const Key, T>}}) is called
with exactly the same arguments as supplied to {{tt|emplace}}, forwarded via
{{c|std::forward<Args>(args)...}}.}}

{{cpp/container/note_iterator_invalidation|{{{1|}}}|emplace}}
```

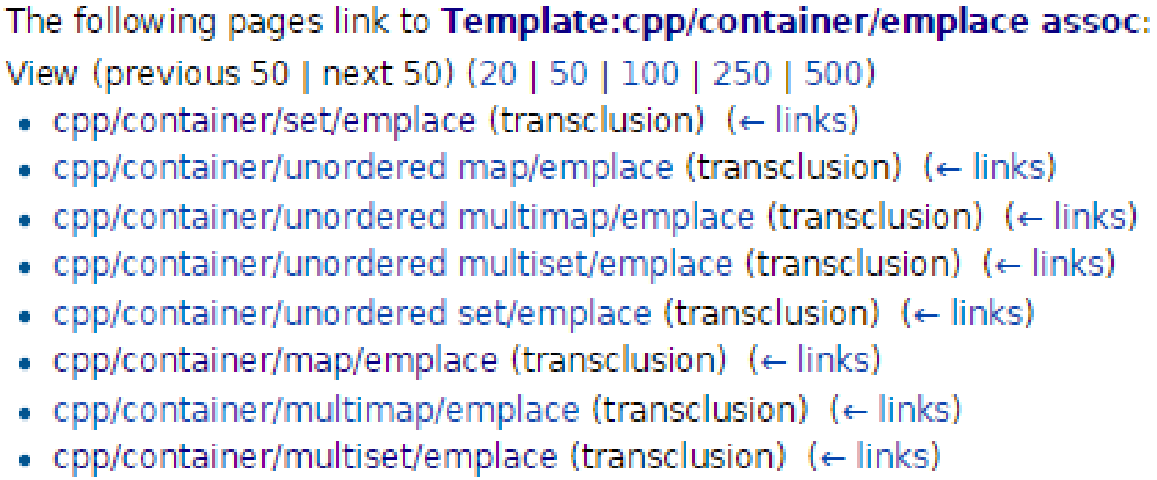
C++ specific conditionals



Support us Recent changes FAQ Offline version

What links here Related changes Upload file Special pages Page information

Privacy policy About cppreference.com Disclaimers

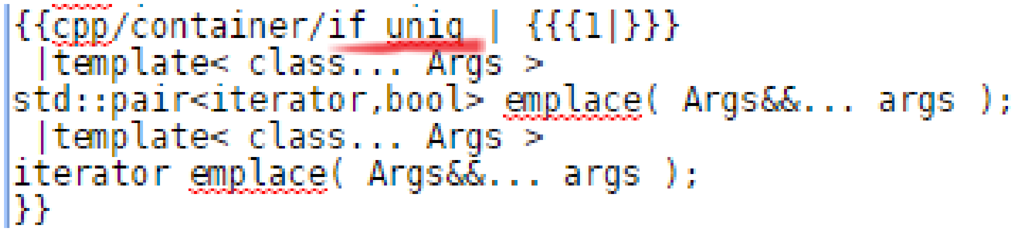


The following pages link to **Template:cpp/container/emplace assoc**:

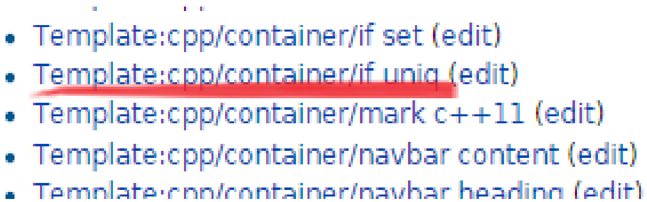
View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)

- cpp/container/set/emplace (transclusion) (← links)
- cpp/container/unordered map/emplace (transclusion) (← links)
- cpp/container/unordered multimap/emplace (transclusion) (← links)
- cpp/container/unordered multiset/emplace (transclusion) (← links)
- cpp/container/unordered set/emplace (transclusion) (← links)
- cpp/container/map/emplace (transclusion) (← links)
- cpp/container/multimap/emplace (transclusion) (← links)
- cpp/container/multiset/emplace (transclusion) (← links)

C++ specific conditionals

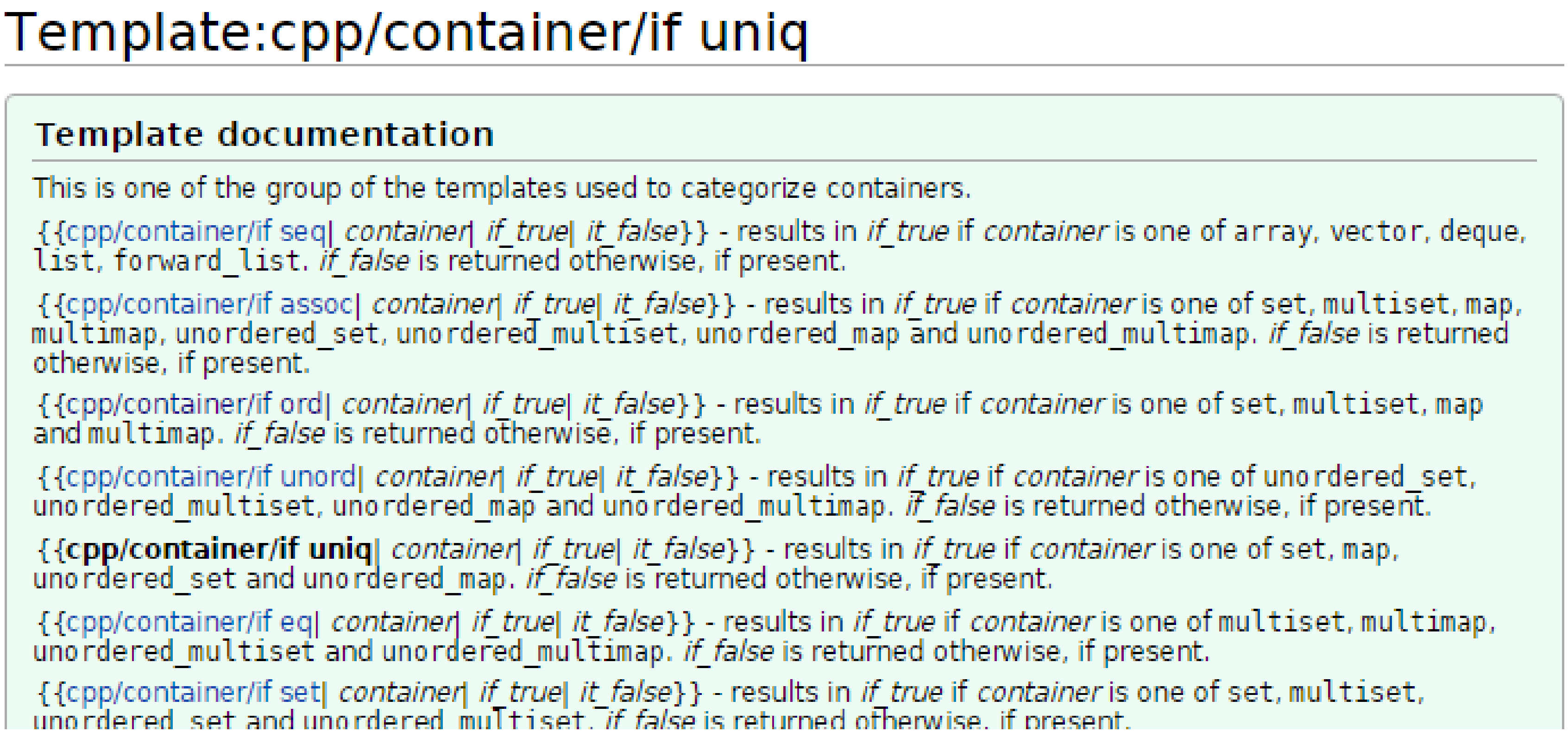


```
{{cpp/container/if uniq | {{{1|}}}
 |template< class... Args >
std::pair<iterator,bool> emplace( Args&&... args );
 |template< class... Args >
iterator emplace( Args&&... args );
}}
```



- Template:cpp/container/if set (edit)
- Template:cpp/container/if uniq (edit)
- Template:cpp/container/mark c++11 (edit)
- Template:cpp/container/navbar content (edit)
- Template:cpp/container/navbar heading (edit)

C++ specific conditionals



Template:cpp/container/if uniq

Template documentation

This is one of the group of the templates used to categorize containers.

{{cpp/container/if seq| *container*| *if_true*| *it_false*}} - results in *if_true* if *container* is one of array, vector, deque, list, forward_list. *if_false* is returned otherwise, if present.

{{cpp/container/if assoc| *container*| *if_true*| *it_false*}} - results in *if_true* if *container* is one of set, multiset, map, multimap, unordered_set, unordered_multiset, unordered_map and unordered_multimap. *if_false* is returned otherwise, if present.

{{cpp/container/if ord| *container*| *if_true*| *it_false*}} - results in *if_true* if *container* is one of set, multiset, map and multimap. *if_false* is returned otherwise, if present.

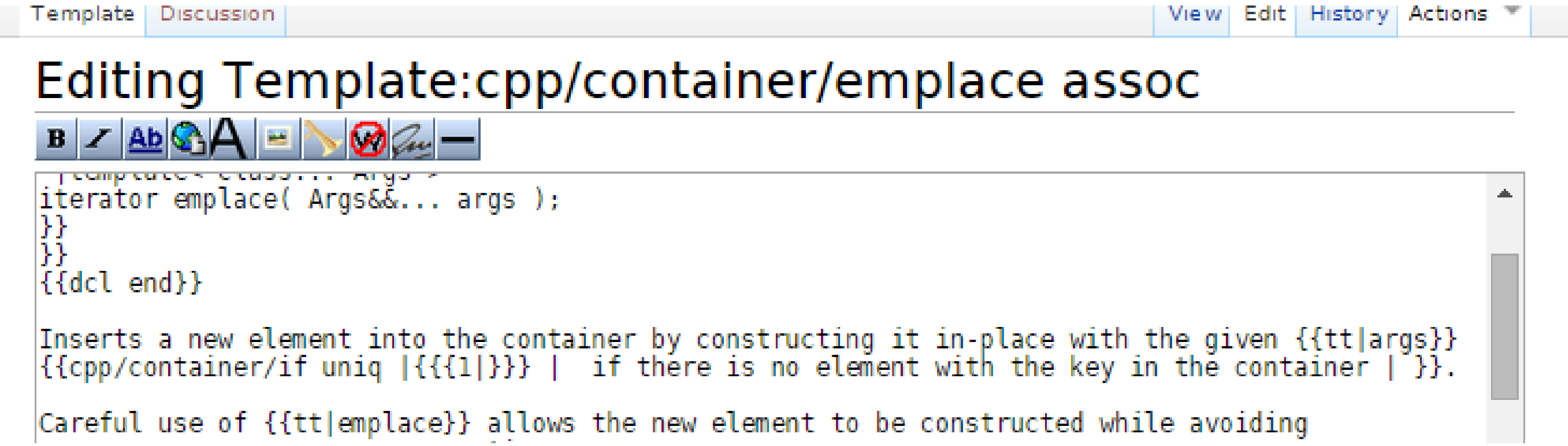
{{cpp/container/if unord| *container*| *if_true*| *it_false*}} - results in *if_true* if *container* is one of unordered_set, unordered_multiset, unordered_map and unordered_multimap. *if_false* is returned otherwise, if present.

{{**cpp/container/if uniq**| *container*| *if_true*| *it_false*}} - results in *if_true* if *container* is one of set, map, unordered_set and unordered_map. *if_false* is returned otherwise, if present.

{{cpp/container/if eq| *container*| *if_true*| *it_false*}} - results in *if_true* if *container* is one of multiset, multimap, unordered_multiset and unordered_multimap. *if_false* is returned otherwise, if present.

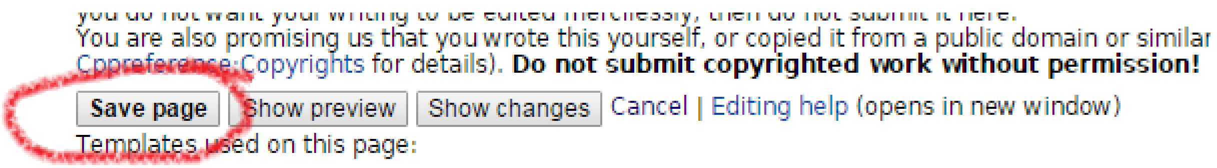
{{cpp/container/if set| *container*| *if_true*| *it_false*}} - results in *if_true* if *container* is one of set, multiset, unordered_set and unordered_multiset. *if_false* is returned otherwise, if present.

C++ specific conditionals



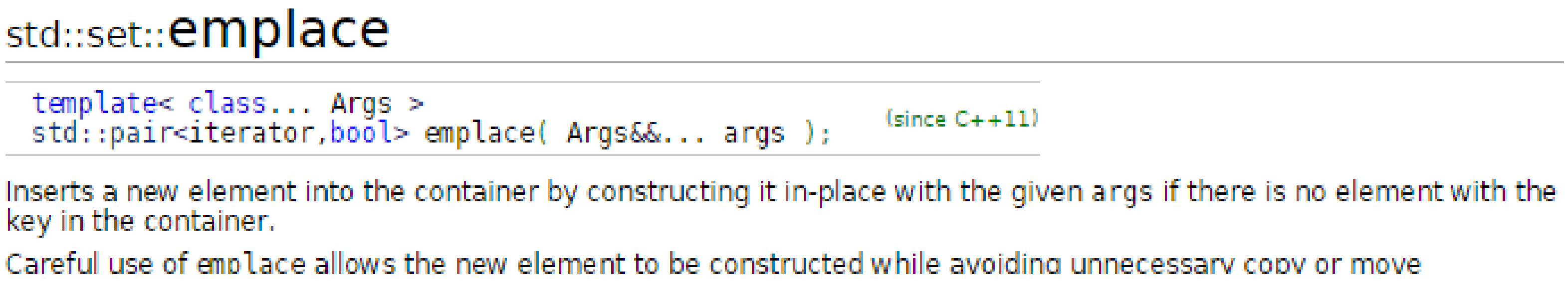
Editing Template:cpp/container/emplace assoc

```
 |template< class... Args >
iterator emplace( Args&&... args );
}}
}}
{{dcl end}}

Inserts a new element into the container by constructing it in-place with the given {{tt|args}}
{{cpp/container/if uniq |{{{1|}}} |  if there is no element with the key in the container | }}.

Careful use of {{tt|emplace}} allows the new element to be constructed while avoiding
```


you do not want your writing to be edited mercilessly, then do not submit it here.
You are also promising us that you wrote this yourself, or copied it from a public domain or similar
Cppreference:Copyrights for details). **Do not submit copyrighted work without permission!**

Templates used on this page:

C++ specific conditionals

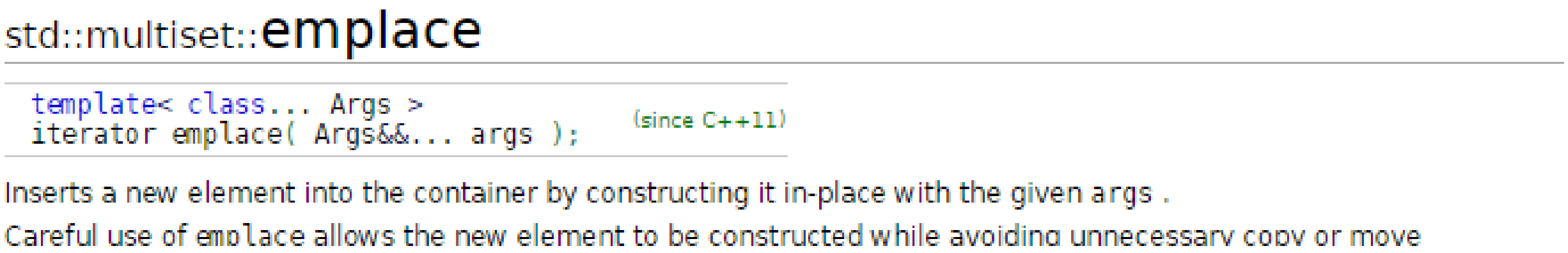


std::set::**emplace**

```
template< class... Args >
std::pair<iterator,bool> emplace( Args&&... args );        (since C++11)
```

Inserts a new element into the container by constructing it in-place with the given args if there is no element with the key in the container.

Careful use of emplace allows the new element to be constructed while avoiding unnecessary copy or move



std::multiset::**emplace**

```
template< class... Args >
iterator emplace( Args&&... args );        (since C++11)
```

Inserts a new element into the container by constructing it in-place with the given args .

Careful use of emplace allows the new element to be constructed while avoiding unnecessary copy or move

Edit without fear

- Use the Talk page
- Always check Preview
- Fill out the Summary
  - Standard chapter/verse is great
  - Stackoverflow discussion URL is even better
- If the page looks “blank”, scroll down!
- When editing templates, use What Links Here

Acknowledgements

- Thanks to
  - User:Nate, for creating and maintaining cppreference
  - User:P12, for creating and maintaining the templates I wouldn’t dare showing here
  - The users who overcame their fear of the templates
  - And the users who didn’t.