

C++

Information

Tutorials

Reference

Articles

Forum

Reference

C library:

Containers:

Input/Output:

Multi-threading:

Other:

<algorithm>

<bitset>

<chrono>

<codecvt>

<complex>

<exception>

<functional>

<initializer_list>

<iterator>

<limits>

<locale>

<memory>

<new>

<numeric>

<random>

<ratio>

<regex>

<stdexcept>

<string>

<system_error>

<tuple>

<typeindex>

<typeinfo>

<type_traits>

<utility>

<valarray>

<string>

class templates:

basic_string

char_traits

classes:

string

u16string

u32string

wstring

functions:

stod

stof

stoi

stol

stold

stoll

stoul

stoull

to_string

to_wstring

string

string::string

string::~string

member functions:

string::append

string::assign

string::at

string::back

string::begin

string::capacity

string::cbegin

string::cend

string::clear

string::compare

string::copy

string::cbegin

liquid

TECHNOLOGIES

C#

XML

Generate C# Object layer from your XML Schema

DOWNLOAD FREE TRIAL

2016

function

relational operators (string)

<string>

C++98

C++14

(1) bool operator==(const string& lhs, const string& rhs);
bool operator==(const char* lhs, const string& rhs);
bool operator==(const string& lhs, const char* rhs);

(2) bool operator!=(const string& lhs, const string& rhs);
bool operator!=(const char* lhs, const string& rhs);
bool operator!=(const string& lhs, const char* rhs);

(3) bool operator<(const string& lhs, const string& rhs);
bool operator<(const char* lhs, const string& rhs);
bool operator<(const string& lhs, const char* rhs);

(4) bool operator<=(const string& lhs, const string& rhs);
bool operator<=(const char* lhs, const string& rhs);
bool operator<=(const string& lhs, const char* rhs);

(5) bool operator>(const string& lhs, const string& rhs);
bool operator>(const char* lhs, const string& rhs);
bool operator>(const string& lhs, const char* rhs);

(6) bool operator>=(const string& lhs, const string& rhs);
bool operator>=(const char* lhs, const string& rhs);
bool operator>=(const string& lhs, const char* rhs);

Relational operators for string

Performs the appropriate comparison operation between the [string](#) objects *lhs* and *rhs*.

The functions use [string::compare](#) for the comparison.

These operators are overloaded in header [<string>](#).

Parameters

lhs, rhs

Arguments to the left- and right-hand side of the operator, respectively.

If of type `char*`, it shall point to a null-terminated character sequence.

Example

1 // string comparisons
2 #include <iostream>
3 #include <vector>
4
5 int main ()
6 {
7 std::string foo = "alpha";
8 std::string bar = "beta";
9
10 if (foo==bar) std::cout << "foo and bar are equal\n";
11 if (foo!=bar) std::cout << "foo and bar are not equal\n";
12 if (foo< bar) std::cout << "foo is less than bar\n";
13 if (foo> bar) std::cout << "foo is greater than bar\n";
14 if (foo<=bar) std::cout << "foo is less than or equal to bar\n";
15 if (foo>=bar) std::cout << "foo is greater than or equal to bar\n";
16
17 return 0;
18 }

Output:

foo and bar are not equal
foo is less than bar
foo is less than or equal to bar


Return Value



true if the condition holds, and false otherwise.

Complexity

Unspecified, but generally up to linear in both *lhs* and *rhs*'s lengths.

<div>string::crend</div> <div>string::c_str</div> <div>string::data</div> <div>string::empty</div> <div>string::end</div> <div>string::erase</div> <div>string::find</div> <div>string::find_first_not_of</div> <div>string::find_first_of</div> <div>string::find_last_not_of</div> <div>string::find_last_of</div> <div>string::front</div> <div>string::get_allocator</div> <div>string::insert</div> <div>string::length</div> <div>string::max_size</div> <div>string::operator+=</div> <div>string::operator=</div> <div>string::operator[]</div> <div>string::pop_back</div> <div>string::push_back</div> <div>string::rbegin</div> <div>string::rend</div> <div>string::replace</div> <div>string::reserve</div> <div>string::resize</div> <div>string::rfind</div> <div>string::shrink_to_fit</div> <div>string::size</div> <div>string::substr</div> <div>string::swap</div> <div>member constants:</div> <div>string::npos</div> <div>non-member overloads:</div> <div>getline (string)</div> <div>operator+ (string)</div> <div>operator<< (string)</div> <div>operator>> (string)</div> <div>relational operators (string)</div> <div>swap (string)</div>
--

 **Graph/Document Database**



Iterator validity

No changes.

Data races

Both objects, *lhs* and *rhs*, are accessed.

Exception safety

C++98

C++14

If an argument of type `char*` does not point to null-terminated character sequence, it causes *undefined behavior*. Otherwise, if an exception is thrown, there are no changes in the `string` (strong guarantee).

See also

string::compare	Compare strings (public member function)
string::find	Find content in string (public member function)
string::operator=	String assignment (public member function)
string::swap	Swap string values (public member function)

[Home page](#) | [Privacy policy](#)

© cplusplus.com, 2000-2016 - All rights reserved - v3.1

[Spotted an error? contact us](#)