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::crbegin

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

```cpp
// string comparisons
#include <iostream>
#include <vector>

int main ()
{
  std::string foo = "alpha";
  std::string bar = "beta";

  if (foo==bar) std::cout << "foo and bar are equal\n";
  if (foo!=bar) std::cout << "foo and bar are not equal\n";
  if (foo< bar) std::cout << "foo is less than bar\n";
  if (foo> bar) std::cout << "foo is greater than bar\n";
  if (foo<=bar) std::cout << "foo is less than or equal to bar\n";
  if (foo>=bar) std::cout << "foo is greater than or equal to bar\n";

  return 0;
}
```

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.

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)