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
string::crend
string::c_str
string::data
string::empty
string::end
string::erase
string::find
string::find_first_not_of

public member function

<string>

std::string::compare

C++98 C++11 C++14

```
      string (1)  int compare (const string& str) const noexcept;
                  int compare (size_t pos, size_t len, const string& str) const;
  substrings (2)  int compare (size_t pos, size_t len, const string& str,
                               size_t subpos, size_t sublen) const;
    c-string (3)  int compare (const char* s) const;
                  int compare (size_t pos, size_t len, const char* s) const;
      buffer (4)  int compare (size_t pos, size_t len, const char* s, size_t n) const;
```

Compare strings

Compares the value of the string object (or a substring) to the sequence of characters specified by its arguments.

The *compared string* is the value of the string object or -if the signature used has a *pos* and a *len* parameters- the substring that begins at its character in position *pos* and spans *len* characters.

This string is compared to a *comparing string*, which is determined by the other arguments passed to the function.

Parameters

`str`
Another string object, used entirely (or partially) as the *comparing string*.

`pos`
Position of the first character in the *compared string*.
If this is greater than the string length, it throws out_of_range.
Note: The first character is denoted by a value of `0` (not `1`).

`len`
Length of *compared string* (if the string is shorter, as many characters as possible).
A value of string::npos indicates all characters until the end of the string.

`subpos, sublen`
Same as *pos* and *len* above, but for the *comparing string*.

`s`
Pointer to an array of characters.
If argument *n* is specified *(4)*, the first *n* characters in the array are used as the *comparing string*.
Otherwise *(3)*, a null-terminated sequence is expected: the length of the sequence with the characters to use as *comparing string* is determined by the first occurrence of a null character.

`n`
Number of characters to compare.

`size_t` is an unsigned integral type (the same as member type `string::size_type`).

Return Value

Returns a signed integral indicating the relation between the strings:

value	relation between *compared string* and *comparing string*
`0`	They compare equal
`<0`	Either the value of the first character that does not match is lower in the *compared string*, or all compared characters match but the *compared string* is shorter.
`>0`	Either the value of the first character that does not match is greater in the *compared string*, or all compared characters match but the *compared string* is longer.

Example

```cpp
// comparing apples with apples
#include <iostream>
#include <string>

int main ()
{
  std::string str1 ("green apple");
  std::string str2 ("red apple");

  if (str1.compare(str2) != 0)
    std::cout << str1 << " is not " << str2 << '\n';

  if (str1.compare(6,5,"apple") == 0)
    std::cout << "still, " << str1 << " is an apple\n";

  if (str2.compare(str2.size()-5,5,"apple") == 0)
    std::cout << "and " << str2 << " is also an apple\n";

  if (str1.compare(6,5,str2,4,5) == 0)
    std::cout << "therefore, both are apples\n";

  return 0;
}
```

Output:

```
green apple is not red apple
still, green apple is an apple
and red apple is also an apple
therefore, both are apples
```

Complexity

Unspecified, but generally up to linear in both the *compared* and *comparing string*'s lengths.

Iterator validity

No changes.

Data races

The object is accessed.

Exception safety

Strong guarantee: if an exception is thrown, there are no changes in the string (except *(1)*, which is guaranteed to not throw).

If s does not point to an array long enough, it causes *undefined behavior*.

If *pos* is greater than the string length, or if *subpos* is greater than *str*'s length, an out_of_range exception is thrown.

See also

string::find	Find content in string (public member function)
string::replace	Replace portion of string (public member function)
string::substr	Generate substring (public member function)
relational operators (string)	Relational operators for string (function)