

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

string::crend

string::c\_str

string::data

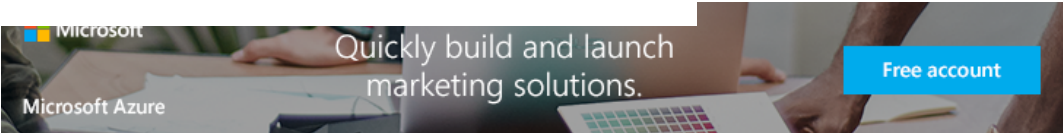
string::empty

string::end

string::erase

string::find

string::find\_first\_not\_of



public member function

std::string::compare

<string>

| C++98 | C++11  | C++14 |
|-------|--|-------|
|       | <div>string (1)</div> <div>substrings (2)</div> <div>c-string (3)</div> <div>buffer (4)</div>  |       |
|       | <div>int compare (const string&amp; str) const noexcept;</div> <div>int compare (size_t pos, size_t len, const string&amp; str) const;</div> <div>int compare (size_t pos, size_t len, const string&amp; str, size_t subpos, size_t sublen) const;</div> <div>int compare (const char* s) const;</div> <div>int compare (size_t pos, size_t len, const char* s) const;</div> <div>int compare (size_t pos, size_t len, const char* s, size_t n) const;</div> |       |

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 <code>string</code> object, used entirely (or partially) as the <i>comparing string</i> .   |
| pos            | Position of the first character in the <i>compared string</i> .<br>If this is greater than the <code>string</code> length, it throws <code>out_of_range</code> .<br>Note: The first character is denoted by a value of 0 (not 1).   |
| len            | Length of <i>compared string</i> (if the string is shorter, as many characters as possible).<br>A value of <code>string::npos</code> indicates all characters until the end of the string.  |
| subpos, sublen | Same as <i>pos</i> and <i>len</i> above, but for the <i>comparing string</i> .  |
| s              | Pointer to an array of characters.<br>If argument <i>n</i> is specified (4), the first <i>n</i> characters in the array are used as the <i>comparing string</i> .<br>Otherwise (3), a null-terminated sequence is expected: the length of the sequence with the characters to use as <i>comparing string</i> 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 <i>compared string</i> and <i>comparing string</i>   |
|-------|---|
| 0     | They compare equal  |
| <0    | Either the value of the first character that does not match is lower in the <i>compared string</i> , or all compared characters match but the <i>compared string</i> is shorter.  |
| >0    | Either the value of the first character that does not match is greater in the <i>compared string</i> , or all compared characters match but the <i>compared string</i> is longer. |

Example

```
1 // comparing apples with apples
2 #include <iostream>
3 #include <string>
4
5 int main ()
6 {
7     std::string str1 ("green apple");
8     std::string str2 ("red apple");
9
10    if (str1.compare(str2) != 0)
11        std::cout << str1 << " is not " << str2 << '\n';
12
13    if (str1.compare(6,5,"apple") == 0)
14        std::cout << "still, " << str1 << " is an apple\n";
15
16    if (str2.compare(str2.size()-5,5,"apple") == 0)
17        std::cout << "and " << str2 << " is also an apple\n";
18
19    if (str1.compare(6,5,str2,4,5) == 0)
20        std::cout << "therefore, both are apples\n";
21
22    return 0;
23 }
```

|                               |
|-------------------------------|
| string::find_first_of         |
| string::find_last_not_of      |
| string::find_last_of          |
| string::front                 |
| string::get_allocator         |
| string::insert                |
| string::length                |
| string::max_size              |
| string::operator+=            |
| string::operator=             |
| string::operator[]            |
| string::pop_back              |
| string::push_back             |
| string::rbegin                |
| string::rend                  |
| string::replace               |
| string::reserve               |
| string::resize                |
| string::rfind                 |
| string::shrink_to_fit         |
| string::size                  |
| string::substr                |
| string::swap                  |
| <b>member constants:</b>      |
| string::npos                  |
| <b>non-member overloads:</b>  |
| getline (string)              |
| operator+ (string)            |
| operator<< (string)           |
| operator>> (string)           |
| relational operators (string) |
| swap (string)                 |

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**

|   |   |
|---|---|
| <a href="#">string::find</a>                  | Find content in string (public member function )    |
| <a href="#">string::replace</a>               | Replace portion of string (public member function ) |
| <a href="#">string::substr</a>                | Generate substring (public member function )        |
| <a href="#">relational operators (string)</a> | Relational operators for string (function )         |