function

std::operator+ (string)

<string>

C++98 C++11

string (1)	`string operator+ (const string& lhs, const string& rhs);`
	`string operator+ (string&&      lhs, string&&      rhs);`
	`string operator+ (string&&      lhs, const string& rhs);`
	`string operator+ (const string& lhs, string&&      rhs);`

c-string (2)	`string operator+ (const string& lhs, const char*   rhs);`
	`string operator+ (string&&      lhs, const char*   rhs);`
	`string operator+ (const char*   lhs, const string& rhs);`
	`string operator+ (const char*   lhs, string&&      rhs);`

character (3)	`string operator+ (const string& lhs, char          rhs);`
	`string operator+ (string&&      lhs, char          rhs);`
	`string operator+ (char          lhs, const string& rhs);`
	`string operator+ (char          lhs, string&&      rhs);`

Concatenate strings

Returns a newly constructed string object with its value being the concatenation of the characters in *lhs* followed by those of *rhs*.

C++11

In the signatures taking at least one *rvalue reference* as argument, the returned object is *move-constructed* by passing this argument, which is left in an unspecified but valid state. If both arguments are *rvalue references*, only one of them is moved (it is unspecified which), with the other one preserving its value.

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  // concatenating strings
2  #include <iostream>
3  #include <string>
4
5  main ()
6  {
7    std::string firstlevel ("com");
8    std::string secondlevel ("cplusplus");
9    std::string scheme ("http://");
10   std::string hostname;
11   std::string url;
12
13   hostname = "www." + secondlevel + '.' + firstlevel;
14   url = scheme + hostname;
15
16   std::cout << url << '\n';
17
18   return 0;
19 }
```

Output:

```
http://www.cplusplus.com
```

Return Value

A string whose value is the concatenation of *lhs* and *rhs*.

Complexity

Unspecified, but generally linear in the resulting string length (and linear in the length of the non-moved argument for signatures with *rvalue references*).

Iterator validity

The signatures with *rvalue references* may invalidate iterators, pointers and references related to the moved string.

Data races

The signatures with *rvalue references* modify the moved string.

Exception safety

Strong guarantee: if an exception is thrown, there are no changes in either string objects.

If s is not a null-terminated character sequence, it causes *undefined behavior*.

If the resulting string length would exceed the max_size, a length_error exception is thrown.
A bad_alloc exception is thrown if the function needs to allocate storage and fails.

See also

string::append	Append to string (public member function)
string::insert	Insert into string (public member function)
string::operator+=	Append to string (public member function)