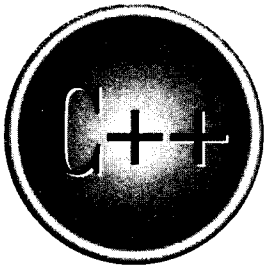
The Complete Reference



Chapter 36

The Standard

This chapter describes the Standard C++ string class. C++ supports character strings two ways. The first is as a null-terminated character array. This is sometimes referred to as a *C string*. The second way is as a class object of type **basic_string**. There are two specializations of **basic_string**: **string**, which supports **char** strings, and **wstring**, which supports **wchar_t** (wide character) strings. Most often, you will use string objects of type **string**.

The **basic_string** class is essentially a container. This means that iterators and the STL algorithms can operate on strings. However, strings have additional capabilities.

A class used by **basic_string** is **char_traits**, which defines several attributes of the characters that comprise a string. It is important to understand that while the most common strings are made up of either **char** or **wchar_t** characters, **basic_string** can operate on any object that can be used to represent a text character. Both **basic_string** and **char_traits** are described here.

Note *For an overview of using the string class, refer to Chapter 24.*

The basic_string Class

The template specification for **basic_string** is

```
template <class CharType, class Attr = char_traits<CharType>,
          class Allocator = allocator<T> > class basic_string
```

Here, **CharType** is the type of character being used, **Attr** is the class that describes the character's traits, and **Allocator** specifies the allocator. **basic_string** has the following constructors:

```
explicit basic_string(const Allocator &a = Allocator( ));
basic_string(size_type len, CharType ch ,
             const Allocator &a = Allocator( ));
basic_string(const CharType *str,  const Allocator &a = Allocator( ));
basic_string(const CharType *str, size_type len,
             const Allocator &a = Allocator( ));
basic_string(const basic_string &str, size_type indx = 0,
             size_type len=npos, const Allocator &a = Allocator( ));
template <class InIter> basic_string(InIter start, InIter end,
             const Allocator &a = Allocator( ));
```

The first form constructs an empty string. The second form constructs a string that has *len* characters of value *ch*. The third form constructs a string that contains the same elements as *str*. The fourth form constructs a string that contains a substring of *str* that begins at zero and is *len* characters long. The fifth form constructs a string from another

basic_string using the substring that begins at *indx* that is *len* characters long. The sixth form constructs a string that contains the elements in the range specified by *start* and *end*.

The following comparison operators are defined for **basic_string**:

==, <, <=, !=, >, >=

Also defined is the + operator, which yields the result of concatenating one string with another, and the I/O operators << and >>, which can be used to input and output strings.

The + operator can be used to concatenate a string object with another string object or a string object with a C-style string. That is, the following variations are supported:

string + string
string + C-string
C-string + string

The + operator can also be used to concatenate a character onto the end of a string.

The **basic_string** class defines the constant **npos**, which is –1. This constant represents the length of the longest possible string.

In the descriptions, the generic type **CharType** represents the type of character stored by a string. Since the names of the placeholder types in a template class are arbitrary, **basic_string** declares **typedef**ed versions of these types. This makes the type names concrete. The types defined by **basic_string** are shown here:

size_type	Some integral type loosely equivalent to **size_t**.
reference	A reference to a character within a string.
const_reference	A **const** reference to a character within a string.
iterator	An iterator.
const_iterator	A **const** iterator.
reverse_iterator	A reverse iterator.
const_reverse_iterator	A **const** reverse iterator.
value_type	The type of character stored in a string.
allocator_type	The type of the allocator.
pointer	A pointer to a character within a string.
const_pointer	A **const** pointer to a character within a string.
traits_type	A **typedef** for **char_traits<CharType>**
difference_type	A type that can store the difference between two addresses.

The member functions defined by **basic_string** are shown in Table 36-1. Since the vast majority of programmers will be using **char** strings (and to keep the descriptions easy-to-understand), the table uses the type **string**, but the functions also apply to objects of type **wstring** (or any other type of **basic_string**).

Member	Description
string &append(const string &*str*);	Appends *str* onto the end of the invoking string. Returns ***this**.
string &append(const string &*str*, size_type *indx*, size_type *len*);	Appends a substring of *str* onto the end of the invoking string. The substring being appended begins at *indx* and runs for *len* characters. Returns ***this**.
string &append(const CharType **str*);	Appends *str* onto the end of the invoking string. Returns ***this**.
string &append(const CharType **str*, size_type *num*);	Appends the first *num* characters from *str* onto the end of the invoking string. Returns ***this**.
string &append(size_type *len*, CharType *ch*);	Appends *len* characters specified by *ch* onto the end of the invoking string. Returns ***this**.
template<class InIter> string &append(InIter *start*, InIter *end*);	Appends the sequence specified by *start* and *end* onto the end of the invoking string. Returns ***this**.
string &assign(const string &*str*);	Assigns *str* to the invoking string. Returns ***this**.
string &assign(const string &*str*, size_type *indx*, size_type *len*);	Assigns a substring of *str* to the invoking string. The substring being assigned begins at *indx* and runs for *len* characters. Returns ***this**.
string &assign(const CharType **str*);	Assigns *str* to the invoking string. Returns ***this**.

Table 36-1. *The String Member Functions*

Member	Description
string &assign(const CharType *str, size_type len);	Assigns the first *len* characters from *str* to the invoking string. Returns ***this**.
string &assign(size_type len, CharType ch);	Assigns *len* characters specified by *ch* to the end of the invoking string. Returns ***this**.
template<class InIter> string &assign(InIter start, InIter end);	Assigns the sequence specified by *start* and *end* to the invoking string. Returns ***this**.
reference at(size_type indx); const_reference at(size_type indx) const;	Returns a reference to the character specified by *indx*.
iterator begin( ); const_iterator begin( ) const;	Returns an iterator to the first element in the string.
const CharType *c_str( ) const;	Returns a pointer to a C-style (i.e., null-terminated) version of the invoking string.
size_type capacity( ) const;	Returns the current capacity of the string. This is the number of characters it can hold before it will need to allocate more memory.
int compare(const string &str) const;	Compares *str* to the invoking string. It returns one of the following: Less than zero if ***this** < *str* Zero if ***this** == *str* Greater than zero if ***this** > *str*
int compare(size_type indx, size_type len, const string &str) const;	Compares *str* to a substring within the invoking string. The substring begins at *indx* and is *len* characters long. It returns one of the following: Less than zero if ***this** < *str* Zero if ***this** == *str* Greater than zero if ***this** > *str*

Table 36-1. *The String Member Functions* (continued)

Member	Description
int compare(size_type *indx*, size_type *len*, const string &*str*, size_type *indx2*, size_type *len2*) const;	Compares a substring of *str* to a substring within the invoking string. The substring in the invoking string begins at *indx* and is *len* characters long. The substring in *str* begins at *indx2* and is *len2* characters long. It returns one of the following: Less than zero if ***this** < *str* Zero if ***this** == *str* Greater than zero if ***this** > *str*
int compare(const CharType **str*) const;	Compares *str* to the invoking string. It returns one of the following: Less than zero if ***this** < *str* Zero if ***this** == *str* Greater than zero if ***this** > *str*
int compare(size_type *indx*, size_type *len*, const CharType **str*, size_type *len2* = npos) const;	Compares a substring of *str* to a substring within the invoking string. The substring in the invoking string begins at *indx* and is *len* characters long. The substring in *str* begins at zero and is *len2* characters long. It returns one of the following: Less than zero if ***this** < *str* Zero if ***this** == *str* Greater than zero if ***this** > *str*
size_type copy(CharType **str*, size_type *len*, size_type *indx* = 0) const;	Beginning at *indx*, copies *len* characters from the invoking string into the character array pointed to by *str*. Returns the number of characters copied.
const CharType *data( ) const;	Returns a pointer to the first character in the invoking string.
bool empty( ) const;	Returns **true** if the invoking string is empty and **false** otherwise.

Table 36-1. *The String Member Functions* (continued)

Member	Description
iterator end( ); const_iterator end( ) const;	Returns an iterator to the end of the string.
iterator erase(iterator *i*);	Removes character pointed to by *i*. Returns an iterator to the character after the one removed.
iterator erase(iterator *start*, iterator *end*);	Removes characters in the range *start* to *end*. Returns an iterator to the character after the last character removed.
string &erase(size_type *indx* = 0, size_type *len* = npos);	Beginning at *indx*, removes *len* characters from the invoking string. Returns ***this**.
size_type find(const string &*str*, size_type *indx* = 0) const;	Returns the index of the first occurrence of *str* within the invoking string. The search begins at index *indx*. **npos** is returned if no match is found.
size_type find(const CharType **str*, size_type *indx* = 0) const;	Returns the index of the first occurrence of *str* within the invoking string. The search begins at index *indx*. **npos** is returned if no match is found.
size_type find(const CharType **str*, size_type *indx*, size_type *len*) const;	Returns the index of the first occurrence of the first *len* characters of *str* within the invoking string. The search begins at index *indx*. **npos** is returned if no match is found.
size_type find(CharType *ch*, size_type *indx* = 0) const;	Returns the index of the first occurrence of *ch* within the invoking string. The search begins at index *indx*. **npos** is returned if no match is found.

Table 36-1. *The String Member Functions* (continued)

Member	Description
size_type find_first_of(const string &*str*, size_type *indx* = 0) const;	Returns the index of the first character within the invoking string that matches any character in *str*. The search begins at index *indx*. **npos** is returned if no match is found.
size_type find_first_of(const CharType **str*, size_type *indx* = 0) const;	Returns the index of the first character within the invoking string that matches any character in *str*. The search begins at index *indx*. **npos** is returned if no match is found.
size_type find_first_of(const CharType **str*, size_type *indx*, size_type *len*) const;	Returns the index of the first character within the invoking string that matches any character in the first *len* characters of *str*. The search begins at index *indx*. **npos** is returned if no match is found.
size_type find_first_of(CharType *ch*, size_type *indx* = 0) const;	Returns the index of the first occurrence of *ch* within the invoking string. The search begins at index *indx*. **npos** is returned if no match is found.
size_type find_first_not_of(const string &*str*, size_type *indx* = 0) const;	Returns the index of the first character within the invoking string that does not match any character in *str*. The search begins at index *indx*. **npos** is returned if no mismatch is found.
size_type find_first_not_of(const CharType **str*, size_type *indx* = 0) const;	Returns the index of the first character within the invoking string that does not match any character in *str*. The search begins at index *indx*. **npos** is returned if no mismatch is found.

Table 36-1. *The String Member Functions* (continued)

Member	Description
size_type find_first_not_of(const CharType *str, size_type *indx*, size_type *len*) const;	Returns the index of the first character within the invoking string that does not match any character in the first *len* characters of *str*. The search begins at index *indx*. **npos** is returned if no mismatch is found.
size_type find_first_not_of(CharType *ch*, size_type *indx* = 0) const;	Returns the index of the first character within the invoking string that does not match *ch*. The search begins at index *indx*. **npos** is returned if no mismatch is found.
size_type find_last_of(const string &*str*, size_type *indx* = npos) const;	Returns the index of the last character within the invoking string that matches any character in *str*. The search ends at index *indx*. **npos** is returned if no match is found.
size_type find_last_of(const CharType *str, size_type *indx* = npos) const;	Returns the index of the last character within the invoking string that matches any character in *str*. The search ends at index *indx*. **npos** is returned if no match is found.
size_type find_last_of(const CharType *str, size_type *indx*, size_type *len*) const;	Returns the index of the last character within the invoking string that matches any character in the first *len* characters of *str*. The search ends at index *indx*. **npos** is returned if no match is found.
size_type find_last_of(CharType *ch*, size_type *indx* = npos) const;	Returns the index of the last occurrence of *ch* within the invoking string. The search ends at index *indx*. **npos** is returned if no match is found.

Table 36-1. *The String Member Functions* (continued)

Member	Description
size_type find_last_not_of(const string &*str*, size_type *indx* = npos) const;	Returns the index of the last character within the invoking string that does not match any character in *str*. The search ends at index *indx*. **npos** is returned if no mismatch is found.
size_type find_last_not_of(const CharType **str*, size_type *indx* = npos) const;	Returns the index of the last character within the invoking string that does not match any character in *str*. The search ends at index *indx*. **npos** is returned if no mismatch is found.
size_type find_last_not_of(const CharType **str*, size_type *indx*, size_type *len*) const;	Returns the index of the last character within the invoking string that does not match any character in the first *len* characters of *str*. The search ends at index *indx*. **npos** is returned if no mismatch is found.
size_type find_last_not_of(CharType *ch*, size_type *indx* = npos) const;	Returns the index of the last character within the invoking string that does not match *ch*. The search ends at index *indx*. **npos** is returned if no mismatch is found.
allocator_type get_allocator( ) const;	Returns the string's allocator.
iterator insert(iterator *i*, const CharType &*ch*);	Inserts *ch* immediately before the character specified by *i*. An iterator to the character is returned.
string &insert(size_type *indx*, const string &*str*);	Inserts *str* into the invoking string at the index specified by *indx*. Returns ***this**.
string &insert(size_type *indx1*, const string &*str*, size_type *indx2*, size_type *len*);	Inserts a substring of *str* into the invoking string at the index specified by *indx1*. The substring begins at *indx2* and is *len* characters long. Returns ***this**.

Table 36-1. *The String Member Functions* (continued)

Member	Description
string &insert(size_type *indx*, const CharType **str*);	Inserts *str* into the invoking string at the index specified by *indx*. Returns ***this**.
string &insert(size_type *indx*, const CharType **str*, size_type *len*);	Inserts the first *len* characters of *str* into the invoking string at the index specified by *indx*. Returns ***this**.
string &insert(size_type *indx*, size_type *len*, CharType *ch*);	Inserts *len* characters of value *ch* into the invoking string at the index specified by *indx*. Returns ***this**.
void insert(iterator *i*, size_type *len*, const CharType &*ch*)	Inserts *len* copies of *ch* immediately before the element specified by *i*.
template <class InIter> void insert(iterator *i*, InIter *start*, InIter *end*);	Inserts the sequence defined by *start* and *end* immediately before the element specified by *i*.
size_type length( ) const;	Returns the number of characters in the string.
size_type max_size( ) const;	Returns the maximum number of characters that the string can hold.
reference operator[ ](size_type *indx*) const; const_reference operator[ ](size_type *indx*) const;	Returns a reference to the character specified by *indx*.

string &operator=(const string &*str*);
string &operator=(const CharType **str*);
string &operator=(CharType *ch*);
	Assigns the specified string or character to the invoking string. Returns ***this**.

string &operator+=(const string &*str*);
string &operator+=(const CharType **str*);
string &operator+=(CharType *ch*);
	Appends the specified string or character onto the end of the invoking string. Returns ***this**.

void push_back (const CharType *ch*)	Adds *ch* to the end of the invoking string.
reverse_iterator rbegin( ); const_reverse_iterator rbegin( ) const;	Returns a reverse iterator to the end of the string.
reverse_iterator rend( ); const_reverse_iterator rend( ) const;	Returns a reverse iterator to the start of the string.

Table 36-1. *The String Member Functions* (continued)

Member	Description
string &replace(size_type *indx*, size_type *len*, const string &*str*);	Replaces up to *len* characters in the invoking string, beginning at *indx* with the string in *str*. Returns ***this**.
string &replace(size_type *indx1*, size_type *len1*, const string &*str*, size_type *indx2*, size_type *len2*);	Replaces up to *len1* characters in the invoking string beginning at *indx1* with the *len2* characters from the string in *str* that begins at *indx2*. Returns ***this**.
string &replace(size_type *indx*, size_type *len*, const CharType **str*);	Replaces up to *len* characters in the invoking string, beginning at *indx* with the string in *str*. Returns ***this**.
string &replace(size_type *indx*, size_type *len1*, const CharType **str*, size_type *len2*);	Replaces up to *len1* characters in the invoking string beginning at *indx* with the *len2* characters from the string in *str*. Returns ***this**.
string &replace(size_type *indx*, size_type *len1*, size_type *len2*, CharType *ch*);	Replaces up to *len1* characters in the invoking string beginning at *indx* with *len2* characters specified by *ch*. Returns ***this**.
string &replace(iterator *start*, iterator *end*, const string &*str*);	Replaces the range specified by *start* and *end* with *str*. Returns ***this**.
string &replace(iterator *start*, iterator *end*, const CharType **str*);	Replaces the range specified by *start* and *end* with *str*. Returns ***this**.
string &replace(iterator *start*, iterator *end*, const CharType **str*, size_type *len*);	Replaces the range specified by *start* and *end* with the first *len* characters from *str*. Returns ***this**.
string &replace(iterator *start*, interator *end*, size_type *len*, CharType *ch*);	Replaces the range specified by *start* and *end* with the *len* characters specified by *ch*. Returns ***this**.

Table 36-1. *The String Member Functions* (continued)

Member	Description
template <class InIter> string &replace(iterator *start1*, interator *end1*, InIter *start2*, InIter *end2*);	Replaces the range specified by *start1* and *end1* with the characters specified by *start2* and *end2*. Returns ***this**.
void reserve(size_type *num* = 0);	Sets the capacity of the string so that it is equal to at least *num*.
void resize(size_type *num*) void resize(size_type *num*, CharType *ch*);	Changes the size of the string to that specified by *num*. If the string must be lengthened, then elements with the value specified by *ch* are added to the end.
size_type rfind(const string &*str*, size_type *indx* = npos) const;	Returns the index of the last occurrence of *str* within the invoking string. The search ends at index *indx*. **npos** is returned if no match is found.
size_type rfind(const CharType **str*, size_type *indx* = npos) const;	Returns the index of the last occurrence of *str* within the invoking string. The search ends at index *indx*. **npos** is returned if no match is found.
size_type rfind(const CharType **str*, size_type *indx*, size_type *len*) const;	Returns the index of the last occurrence of the first *len* characters of *str* within the invoking string. The search ends at index *indx*. **npos** is returned if no match is found.
size_type rfind(CharType *ch*, size_type *indx* = npos) const;	Returns the index of the last occurrence of *ch* within the invoking string. The search ends at index *indx*. **npos** is returned if no match is found.
size_type size( ) const;	Returns the number of characters currently in the string.

Table 36-1. *The String Member Functions* (continued)

Member	Description
string substr(size_type *indx* = 0, size_type *len* = npos) const;	Returns a substring of *len* characters beginning at *indx* within the invoking string.
void swap(string &*str*)	Exchanges the characters stored in the invoking string with those in *ob*.

Table 36-1. *The String Member Functions* (continued)

The char_traits Class

The class **char_traits** describes several attributes associated with a character. Its template specification is shown here:

```
template<class CharType> struct char_traits
```

Here, **CharType** specifies the type of the character.

The C++ library provides two specializations of **char_traits**: one for **char** characters and one for **wchar_t** characters. The **char_traits** class defines the following five data types:

char_type	The type of the character. This is a **typedef** for **CharType**.
int_type	An integer type that can hold a character of type **char_type** or the EOF character.
off_type	An integer type that can represent an offset in a stream.
pos_type	An integer type that can represent a position in a stream.
state_type	An object type that stores the conversion state. (Applies to multibyte characters.)

The member functions of **char_traits** are shown in Table 36-2.

Member	Description
static void assign(char_type &*ch1*, const char_type &*ch2*);	Assigns *ch2* to *ch1*.
static char_type *assign(char_type **str*, size_t *num*, char_type *ch2*);	Assigns *ch2* to the first *num* characters in *str*. Returns *str*.
static int compare(const char_type **str1*, const char_type **str2*, size_t *num*);	Compares *num* characters in *str1* to those in *str2*. Returns zero if the strings are same. Otherwise, returns less than zero if *str1* is less than *str2* or greater than zero if *str1* is greater than *str2*.
static char_type *copy(char_type **to*, const char_type **from*, size_t *num*);	Copies *num* characters from *from* to *to*. Returns *to*.
static int_type eof( );	Returns the end-of-file character.
static bool eq(const char_type &*ch1*, const char_type &*ch2*);	Compares *ch1* to *ch2* and returns **true** if the characters are the same and **false** otherwise.
static bool eq_int_type(const int_type &*ch1*, const int_type &*ch2*);	Returns **true** if *ch1* equals *ch2* and **false** otherwise.
static const char_type *find(const char_type **str*, size_t *num*, const char_type **ch*);	Returns a pointer to the first occurrence of *ch* in *str*. Only the first *num* characters are examined. Returns a null pointer on failure.
static size_t length(const char_type **str*);	Returns the length of *str*.
static bool lt(const char_type &*ch1*, const char_type &*ch2*);	Returns **true** if *ch1* is less than *ch2* and **false** otherwise.

Table 36-2. *The **char_traits** Member Functions*

Member	Description
static char_type *move(char_type *to, const char_type *from, size_t num);	Copies *num* characters from *from* to *to*. Returns *to*.
static int_type not_eof(const int_type &ch);	If *ch* is not the EOF character, then *ch* is returned. Otherwise, the EOF character is returned.
static char_type to_char_type(const int_type &ch);	Converts *ch* into a **char_type** and returns the result.
static int_type to_int_type(const char_type &ch);	Converts *ch* into an **int_type** and returns the result.

Table 36-2. *The* ***char_traits*** *Member Functions* (continued)