---

# Chapter 13. Strings

This chapter presents the string types of the C++ standard library. It describes the basic class template `basic_string<>` and its standard specializations `string`, `wstring`, `u16string`, and `u32string`.

Strings can be a source of confusion because it is not clear what the term *string* means. Does it mean an ordinary character array of type `char*` (with or without the `const` qualifier)? Is it an instance of class `string<>`? Or is it a general name for objects that are kinds of strings? In this chapter, I use the term *string* for objects of one of the string types in the C++ standard library: `string`, `wstring`, `u16string`, or `u32string`. For "ordinary strings" of type `char*` or `const char*`, I use the term *C-string*.

Note that with C++98 the type of string literals (such as `"hello"`) was changed into `const char*`. However, to provide backward compatibility, there is an implicit but deprecated conversion to `char*` for them. Strictly speaking, the original type of a literal such as `"hello"` is `const char[6]`. But this type automatically converts (*decays*)to `const char*`, so you can almost always use (and see) `const char*` in declarations. Nevertheless, when working with templates, the difference might matter because for reference template parameters, decay doesn't occur unless type trait `std::decay()` is used (see Section 5.4.2, page 132).

#### Recent Changes with C++11

C++98 specified almost all features of the string classes. Here is a list of the most important features added with C++11:

- Strings now provide `front()` and `back()` to access the first or last element (see Section 13.2.6, page 671) and `shrink_to_fit()` to shrink capacity (see Section 13.2.5, page 670).
- Strings now provide convenience functions to convert strings to numeric values and vice versa (see Section 13.2.13, page 681).
- `data()` and `c_str()` no longer invalidate references, iterators, and pointers to strings (see Section 13.2.6, page 672).
- Strings now support move semantics (see Section 13.2.9, page 676) and initializer lists (see Section 13.2.8, page 675).
- Besides `string` and `wstring`, the `basic_string<>` specializations `u16string` and `u32string` are predefined now (see Section 13.2.1, page 664).
- Strings Strings are now indirectly required to provide an *end-of-string* character (`'\0'` for `string`) because for a string `s`, `s[s.length()]` is always valid and `s.data()` returns the characters including a trailing *end-of-string* character (see Section 13.1.2, page 662).
- Reference-counted implementations of string classes are no longer supported (see Section 13.2.16, page 692).