■ StackExchange ▼

sign up log in tour help ▼ stack overflow careers

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

Take the 2-minute tour

## Difference between a C++ string and a C-string (.c\_str()) [duplicate]

## Track your code, bugs, and work. Anywhere. For free.



Possible Duplicates: Why does std::string not provide a conversion to const char\*? Why doesn't std::string provide implicit conversion to char\*?

```
case 1 :
void readFile ( const string& inputfile ) {
    ifstream in ( inputfile );
}

case 2:
void readFile ( const string& inputfile ) {
    ifstream in ( inputfile . c_str() );
}
```

Of course, I know how I can call ifstream with a required parameter, but what is the real difference between a C++ string and a null-terminated sequence of characters (C-string) .c\_str() ?\*\*

I think automatic type conversion should be do its job, that is, automatically convert a C++ string to .c\_str(). Am I wrong?

Case 1 gives an error, and case 2 works fine. Is it possible to convert case 1 to case 2 over using static\_cast<>?

C++ C

edited May 21 '12 at 21:00





marked as duplicate by mu is too short, Mike Seymour, Tony D, Aamir, Loki Astari Apr 21 '11 at 8:30

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

- 2 Duplicates: stackoverflow.com/questions/492061/... and stackoverflow.com/questions/4096210/... Max Lybbert Apr 21 '11 at 6:38
- 1 the file stream constructors used to take only a const char\* so you had to call c\_str() if you were using string. however, in the new C++0x standard they have fixed that, so case 1 will be valid – Marius Bancila Apr 21 '11 at 7:26

## 3 Answers

What is the real difference between a C++ string and a null-terminated sequence of characters (C-string) .c\_str() ?

A C++ std::string object encapsulates:

- a char array storing the semantic (presumably textual) value
  - some implementations store short text strings directly in the std::string object
  - · otherwise heap memory is typically used to store the actual string content
- a pointer (possibly via some other control structure) to the character array
- std::string::size\_type
   variables recording the size and capacity of the string
- possibly other things

In practice, the std::string 's textual data - whether internally buffered or kept on the heap, is overwhelmingly likely in real-world implementations to be stored as a C-string ASCIIZ value, such that c\_str() can trivially return it's address, but that's not required by the Standard. A nearworst-case (just within the boundaries of credibility) scenario is that the string has a second pointer, and c str() copies the non-NUL-terminated string content into a newly allocated heap area that it NUL terminates. The only time this would seem beneficial is if the NUL itself tipped the string over some capacity boundary, such as from an short-string optimisation / internal buffer to heap, or from 1 page of heap memory to 2, 2 to 3, etc...

I think automatic type conversion should be do its job, that is, automatically convert a C++ string to .c str(). Am I wrong?

Yes it can do it, but not safely (see linked possible-dupe questions).

Case 1 divise an error, and case 2 works fine is it nossible to convert case 1 to case 2 over THIS PAGE ISSAFE VAULT ISACCESSSITE ISINFOBAR IGNOREDOPENCLOSEDVAULTHIDDEN

static\_cast<> can't convert a std::string object to a const char\* ... remember the string object itself has all those other things in, and typically (always for all but the smallest of strings) only has a pointer to the actual textual data.

edited May 21 '12 at 21:02

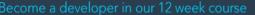


8.207 • 10 • 57 • 94





## Tired of Self-Taught Coding?





Learn More

There's no conversion of std::string to const char\*, so you have to call .c\_str() - no explicit cast will help you here.

answered Apr 21 '11 at 6:37



98.9k • 29 • 243 • 601

std::string is a class that stores a string, and happens to be the standard way to handling strings in C++, how exactly it should be implemented and how it should use memory is not explicitly defined by the C++ standard, it only defines it API. Unfortunately std::ifstream came before the standardization of std::string, so its interface uses the old C string.

C strings are just char \* (pointers to char ) that follows certain conventions imposed by the C standard. Namely, when the data pointed by the pointer happens to end with '\@', it can be considered a string. Thus, if I do not use char \* as the convention says (by not following my relevant data with a '\o'), it can not be considered a C string, even if it is of the right type. Using such pointer in standard functions that expect C strings is certainly an error.

Calling .c\_str() will give a const char \* pointer that is usable as a C string, as the C standard defines it. The pointer returned by .c\_str() is not the same as the pointer to the std::string object, nor the C++ standard requires it to be so, but the returned C string belongs and is managed by the C++ object.

answered Apr 21 '11 at 7:05



**3,662** • 1 • 19 • 47