Why are there two different getline() functions (if indeed there are)?

Every time I do a quick snippet of C++ code line

```
std::string s;
cin >> s;
```

I curse myself because I forgot it stops at the whitespace rather than getting an entire line.

Then, on remembering `getline`, I invariably become confused as to the two varieties:

```
std::string s;
getline (std::cin, s);
```

and:

```
char cs[256];
std::cin.getline (cs, sizeof (cs));
```

Is there actually a difference between these two other than the data type?

It seems to me the C++ way should be the former. Under what circumstances would I use the latter, given that I probably should be using real strings instead of null-terminated character arrays anyway?

And, since input should really be the purview of the input streams, why isn't the former part of `istream`?

c++ string getline

asked Feb 2 '11 at 8:48

paxdiablo
432k 105 833 1299

5 Answers

Bear in mind that the Standard library is composed from 3 (main) parts: IOStream, String and STL, plus some tossed in goodies and the C-headers.

I don't see anything weird in having those parts loosely coupled (though I wish it was not the case).

Other incongruities include: `std::string::length` vs `std::string::size`, the latter having been added for interface compatibility with the STL and the former having been retained for compatibility with older code.

edited Feb 2 '11 at 10:28

answered Feb 2 '11 at 9:14

Matthieu M.
125k 13 144 314

3 I think the phrase is "loosely coupled", "poorly integrated" has negative connotations. I think that it is a good thing (tm) that you can use `std::string` without needing `std::istream` and use `std::istream` without needing `std::string`. – Charles Bailey Feb 2 '11 at 9:32

2 @Charles: would you believe me if I told you that I made the exact opposite change before hitting "submit" :) ? I would personally have favored a more "uniform" standard library. The C++ standard library is the epitome of "organic growth". – Matthieu M. Feb 2 '11 at 10:30

1 @Charles: I definitely think that it's wrong that iostreams and string aren't coupled. I mean, iostream depends on manipulating strings, at a basic level. Reducing dependencies is one thing, providing member functions that work with C-string is inexcusable if you're not the string class. – Puppy Feb 2 '11 at 10:42

@Puppy: I'd definitely disagree that iostream depends on manipulating strings, but would prefer if they were more tightly integrated. – Mooing Duck Jul 15 '14 at 23:53

The global getline() function works with C++ std::string objects.

The istream::getline() methods work with "classic" C strings (pointers to `char`).

answered Feb 2 '11 at 8:57

Frédéric Hamidi
147k 19 245 312

So, if the only thing it inputs is strings, why isn't it `s.getline (std::cin)` instead of breaking my lovely `object.method()` view of the world? – paxdiablo Feb 2 '11 at 9:00

1 @paxdiablo, it might be a question of semantics: `getline()` is a stream operation, not a string operation, so it would be strange to make it a method of the `std::string` class. – Frédéric Hamidi Feb 2 '11 at 9:03

2 My guess is <iostream> doesn't know about <string>, therefore std::cin can't have method that returns std::string. – Grozz Feb 2 '11 at 9:12

@Grozz, indeed, the two classes are not coupled. – Frédéric Hamidi Feb 2 '11 at 9:13

It's a common interface-design problem. `cin.getline()` is a natural way to make the request, but to avoid making the stream code dependent on `<string>`, no `cin.getline(std::string&)` function can be offered. The free-standing `getline(cin, s)` can later be added once strings have been brought into scope. Not a problem for `char*` as there's nothing to `#include` - all part of the language anyway.

In some ways, it's nice when languages allow later code to add further functions to existing classes (e.g. Ruby), but in other ways the delocalisation bites and, well, compromises maintainability. Then of course there's the popular argument for minimal member functions and lots of free-standing functions: I personally think it shouldn't be taken so far as to make the interface less intuitive and expressive, but each to their own.

edited Jul 15 '14 at 23:35

anthropomorphic
1,750 1 18 39

answered Feb 2 '11 at 9:25

Tony D
63.9k 7 73 132

The `getline` variant inside the iostreams library does not support strings as targets, so the string library defined a variant that does.

answered Feb 2 '11 at 8:57

Simon Richter
14.4k 1 20 40

Yes the modern C++ way is to use the free function and input an std::string.

But IOStream has a far longer history (the standard version is at least the third incarnation of the same design) than std::string and that history explain why things are the way they are.

(The getline member has the advantage that it doesn't imply dynamic allocation; that characteristic can be handy at time but would probably not be enough to justify it in a from scratch design).

answered Feb 2 '11 at 9:08

AProgrammer
34.8k 5 50 105