C++ cast syntax styles

Asked 16 years, 3 months ago Modified 8 years ago Viewed 20k times

40

A question related to Regular cast vs. static_cast vs. dynamic_cast:

What cast syntax style do you prefer in C++?

- C-style cast syntax: `(int)foo`
- C++-style cast syntax: `static_cast<int>(foo)`
- constructor syntax: `int(foo)`

They may not translate to exactly the same instructions (do they?) but their effect should be the same (right?).

If you're just casting between the built-in numeric types, I find C++-style cast syntax too verbose. As a former Java coder I tend to use C-style cast syntax instead, but my local C++ guru insists on using constructor syntax.

What do you think?

c++ coding-style casting

Share

Improve this question

Follow

edited May 23, 2017 at 11:47

Community Bot

1 • 1

asked Aug 28, 2008 at 13:01

palm3D

8,220 ● 6 ● 29 ● 33

1 Does this answer your question? When should static_cast, dynamic_cast, const_cast and reinterpret_cast be used? – SuperStormer Jan 2, 2022 at 7:28

10 Answers

Sorted by: Highest score (default)

62

It's best practice *never* to use C-style casts for three main reasons:

- as already mentioned, no checking is performed here. The programmer simply cannot know which of the various casts is used which weakens strong typing
- the new casts are intentionally visually striking. Since casts often reveal a weakness in the code, it's argued that making casts visible in the code is a good thing.
- this is especially true if searching for casts with an automated tool. Finding C-style casts reliably is nearly impossible.

As palm3D noted:

> I find C++-style cast syntax too verbose.

This is intentional, for the reasons given above.

The constructor syntax (official name: function-style cast) is semantically *the same* as the C-style cast and should be avoided as well (except for variable initializations on declaration), for the same reasons. It is debatable whether this should be true even for types that define custom constructors but in Effective C++, Meyers argues that even in those cases you should refrain from using them. To illustrate:

```
void f(auto_ptr<int> x);

f(static_cast<auto_ptr<int> >(new int(5))); // GOOD
f(auto_ptr<int>(new int(5));                // BAD
```

The `static_cast` here will actually call the `auto_ptr` constructor.

Share Improve this answer

Follow

edited Sep 28, 2015 at 18:08

Andrew

5,352 ● 1 ● 24 ● 41

answered Aug 28, 2008 at 13:16

Konrad Rudolph

545k ● 139 ● 956 ● 1.2k

15 I wonder how many times have you searched for a cast in your code with an automated tool... – Blindy Dec 5, 2010 at 5:08

2 @Blindly: it happens. I have already done that. Remember that in C++, unlike some other languages (Java, C#), you can usually program without casts. Every explicit cast in your

code is a potential design flaw. Identifying casts in your C++ code is an important step in refactoring. In C# it would of course be ridiculous to search for casts in code – they're everywhere! – Konrad Rudolph Dec 5, 2010 at 10:30

4 There are two problems with your answer: 1) you mention "two main reasons" but you list three. :) +1 – augustin Sep 22, 2015 at 1:42

7 Isn't the `// GOOD` actually nonsense here? It'd be quite awful to write something like `static_cast<std::string>("hello")` instead of `std::string("hello")` or any similar construction of object of user type. – Ruslan Jan 8, 2016 at 13:47

3 Then someone should have no problem citing precisely where and with what wording (a) Sutter and (b) the rest of the "several C++ authorities" said anything of the sort, because it sounds like (i) news and (ii) nonsense to me. – underscore_d Sep 12, 2018 at 19:03

16

According to Stroustrup:

> The "new-style casts" were introduced to give programmers a chance to state their intentions more clearly and for the compiler to catch more errors.

So really, its for safety as it does extra compile-time checking.

Share Improve this answer

Follow

answered Aug 28, 2008 at 13:07

hometoast

11.8k ● 5 ● 42 ● 58

8

Regarding this subject, I'm following the recommandations made by Scott Meyers (More Effective C++, Item 2 : Prefer C++-style casts).

I agree that C++ style cast are verbose, but that's what I like about them : they are very easy to spot, and they make the code easier to read (which is more important than writing).

They also force you to think about what kind of cast you need, and to chose the right one, reducing the risk of mistakes. They will also help you detecting errors at compile time instead at runtime.

Share Improve this answer

Follow

answered Aug 28, 2008 at 13:19

Jérôme

27k ● 31 ● 101 ● 121

4

Definitely C++-style. The extra typing will help prevent you from casting when you shouldn't :-)

Share Improve this answer

Follow

answered Aug 28, 2008 at 13:17

Ben Collins

20.7k ● 18 ● 128 ● 190

I use static_cast for two reasons.

3

1. It's explicitly clear what's taking place. I can't read over that without realizing there's a cast going on. With C-style casts you eye can pass right over it without pause.
2. It's easy to search for every place in my code where I'm casting.

Share Improve this answer

Follow

answered Aug 28, 2008 at 13:16

Bill the Lizard
405k ● 211 ● 572 ● 889

3

The constructor syntax. C++ is OO, constructors exist, I use them. If you feel the need to annotate these conversion ctor's you should do it for every type, not just the built-in ones. Maybe you use the 'explicit' keyword for conversion ctors but the client syntax mimics exactly what the ctor syntax for built-in types does. Being greppable, that may be true, but what a big surprise that typing more characters makes searches easy. Why treat these ones as special? If you are writing math formulas with lots of int/unsigned/... to and from double/float - graphics - and you need to write a static_cast every time, the look of the formula gets cluttered and is very much unreadable. And it's an uphill battle anyway as a lot of times you will convert without even noticing that you are. For downcasting pointers I do use the static_cast as of course no ctor exists by default that would do that.

Share Improve this answer

answered Dec 23, 2016 at 14:48

Follow

QBziZ
3,250 ● 26 ● 24

1

C-style cast syntax, do not error check. C++-style cast syntax, does some checking. When using static_cast, even if it doesn't do checking, at least you know you should be carefull here.

Share Improve this answer

Follow

answered Aug 28, 2008 at 13:04

CiNN
9,870 ● 6 ● 45 ● 58

1 `static_cast` always checks that the source and destination types are compatible. (It can't protect users against their mistake if they convert a base to a derived type that it doesn't really have, but that's their fault.)
– underscore_d May 12, 2017 at 23:42

1

C-style cast is the worst way to go. It's harder to see, ungreppable, conflates different actions that should not be conflated, and can't do everything that C++-style casts can do. They really should have removed C-style casts from the language.

Share Improve this answer

Follow

answered Aug 28, 2008 at 13:11

DrPizza
18.3k ● 7 ● 42 ● 53

We currently use C-style casts everywhere. I asked the other casting question, and I now see the advantage of using static_cast instead, if for no other reason than it's "greppable" (I like that term). I will probably start using that.

I don't like the C++ style; it looks too much like a function call.

Share Improve this answer

Follow

edited May 23, 2017 at 11:47

Community Bot

1 • 1

answered Aug 28, 2008 at 13:30

Graeme Perrow

57.2k • 24 • 86 • 125

1 looking like a function call can be nice, it allows you to have utility functions which share the same style such as the common `lexical_cast` for converting from strings <-> numeric types. But that's just an opinion. – Evan Teran Apr 4, 2010 at 5:06

Go for C++ style and, at worse, the ugly verbose code snippets that comprised C++'s explicit typecast will be a constant reminder of what we all know (i.e explicit casting is bad -- the lead to the coin-ing of expletives). Do not go with C++ style if you want to master the art of tracking runtime errors.

Share Improve this answer

Follow

answered Feb 26, 2012 at 21:18

CAH

11 ● 1