Would there be any drawback in introducing students to `std::addressof` together with `operator&`? [closed]

Asked 10 years, 4 months ago Modified 10 years, 4 months ago Viewed 198 times

1

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 10 years ago.

Improve this question

I'm a TA in an introductory course to programming, where we teach C++11. In particular, my part is about the basics of memory management (value vs. reference semantics, storage duration, automatic vs. dynamic objects, etc.).

Quite predictably, students find this part to be quite though (for the vast majority of them, this is the first time they hear about this stuff). One of the main issues they have to face is learning how to distinguish between the different meanings of the ampersand (defining a lvalue reference vs. taking the address of a lvalue). To this end, since we are using C++11, I'm wondering if it could make sense to present the `std::addressof` function.

This could help removing any ambiguity. The following snippet:

```
int b = 10;
int* a = &b;
int& c = b;
```

could become:

```
int* a = std::addressof(b);
int& c = b;
```

and the `&` would be associated with exactly one meaning: defining lvalue references. From one side it seems reasonable, since:

- the focus is not on training C++ programmers (more advanced courses switch to Java), and
- I can still present the address-of operator along with `std::addressof`.

The downside I see is teaching something that is not idiomatic. This could be fine given the first bullet on the list. Howerver, I'm mostly worried about **technical** downsides that I cannot foresee at the moment (e.g., run-time costs).

`c++` `c++11`

Share

Improve this question

Follow

edited Jul 30, 2014 at 8:41

asked Jul 29, 2014 at 17:56

Ilio Catallo

3,180 ● 2 ● 24 ● 40

4 If you are teaching people who have no idea what a pointer is, then it's probably not yet time for you to worry about them writing inefficient code. So there's no problem with "runtime costs". Teaching something that is not idiomatic is a much tougher problem. I'm sorry if memory management is hard, but so are other things taught in college, so eventually, they will have to get used to it. – The Paramagnetic Croissant Jul 29, 2014 at 17:59

2 Classes that overwrite `operator&` are broken should should not be used because of that fact. – Kijewski Jul 29, 2014 at 18:01

11 There is no cost, but the students will end up reading code that uses `&` soon enough and they will be confused if you didn't teach them what that means. – Marc Glisse Jul 29, 2014 at 18:01

5 What is this university where you are teaching C++11 are you from the future – user703016 Jul 29, 2014 at 18:02

4 @jalf anyone who has to RAII manage resources returned via parameter. It's quite common on windows due to COM. – Mgetz Jul 29, 2014 at 18:16

3 Answers

Sorted by: Highest score (default)

10

Forget about run-time costs. You should be focused instead about teaching *correct* methods. Using `addressof` is not idiomatic if all you are doing is taking the address of a simple variable. What *is* idiomatic is something like `int* a = &b;` Your students need to understand what `&` means in different contexts.

I appreciate that your "focus is not on training C++ programmers," but I'd implore you to consider the long-term. In your school, I imagine that most of the students who will eventually go on to be C++ programmers in the real world and end up working for someone like me will travel through your class at the beginning. I can't tell you how

many times I have had to go to great lengths to retrain recent graduates because what they learned in school was just flat wrong. Bad habits that you teach them now will tend to stick far more often than you may realize. Even if you fully expect these students to be re-trained in a more advanced class, they will often retain their early lessons. Please don't teach them things now that I will just have to un-teach them later.

Learning about pointers for the first time is **hard**. There is no doubt about that. Most people struggle with it. My recommendation to you is not to try to make it easier by skirting around it with `addressof` or other clever tricks, but to be patient and consistent in your teachings.

Share Improve this answer Follow

answered Jul 29, 2014 at 18:21

John Dibling
101k ● 32 ● 191 ● 331

0

To be honest your curriculum is backwards. Why are you teaching C++11 as an introductory course to students who are going to move on to Java in the more "advanced" courses?

I could see a point to wanting to teach languages "close to the metal" to build an understanding of what a particular piece of Java code means for the computer, but then C would probably be a better choice than C++.

As for the technical downsides, there could be the one that debugging the code will have you step into addressof() all the time. You don't need to worry about performance though; any C++ compiler will be able to inline addressof().

Share Improve this answer Follow

answered Jul 29, 2014 at 18:22

flodin
5,244 ● 4 ● 27 ● 40

@black why is it meaningless? – flodin Jul 29, 2014 at 18:34

2 C++11 is pretty decently beginner-friendly if taught right, and C is no closer to the metal than C++. – chris Jul 29, 2014 at 19:54

I tend to agree with Herb Sutter when he says that C++11/14 can be as simple as Python. For an introductory course, IMHO it is sufficient to present stack-allocated objects, and data structures/algorithms coming from the STL. Do you want to move to the next element in a container? use `std::next` and `std::prev`. Do you want to write a function that takes an iterator? say that you want an iterator (whatever it is) with the signature `template<typename Iterator> f(Iterator i)`. If you want to use reference semantics, use lvalue references, and only at the very end you should present pointers. – Ilio Catallo Jul 30, 2014 at 8:21

@flodin I said it's meaninless (and I didn't remove the message) because your concept that Java is the "dream" of each is programmer is *wrong*. Java is, and lots of students will realize that, often overrated. BTW, I'm not going to start a language war. – edmz Jul 30, 2014 at 9:07

1 I think the rationale behind using Java is that it is a good language if you want to solely focus on the object-oriented paradigm, which is something we tend to do in our software engineering courses. Moreover, Java can be used as the common language between courses, as we use it in the web technologies courses (JEE), and in the mobile development course (Android). – Ilio Catallo Jul 30, 2014 at 9:32

0

In my opinion it would indeed be simpler to deal with `std::addressof` instead of `&`. However, students will need to learn the different meanings of symbols anyway, because they are likely to encounter it in production aka real world code. So although it's tempting, I would advise against using it in teaching.

Even with `std::addressof` instead of `&`, the `&` still has far more meaning in other contexts: It could be a reference, so regarding types both `&` and `&&` are meaningful. Additionally, `&` and `&&` have special meaning for integers and booleans (bitwise and / and). For classes, those operators can be overloaded with various types.

Concerning runtime cost: The template

```
template <typename T>
T* addr(T& t) { return &t; }
```

does not impose runtime overhead, unless it's a terrible compiler. This is because this will most definitely be inlined.

My suggestion for beginners would be to stop worrying about addresses. Pointers are necessary for everything, but one can hide it in the beginning. Learning C++ shouldn't begin with pointers, in my opinion.

Share Improve this answer Follow

answered Jul 29, 2014 at 18:32

stefan

10.3k ● 4 ● 53 ● 91

"*Pointers are necessary for everything, but one can hide it in the beginning. Learning C++ shouldn't begin with pointers, in my opinion.*" I strongly agree with this sentiment, but to be fair, the OP did explicitly mention "*I'm a TA in an introductory course to programming ... the focus is not on training C++ programmers (more advanced courses switch to Java)*". – ildjarn Jul 29, 2014 at 19:44

@ildjarn Would you care to say *why*? It's dependent on the subject of course. If it's about primitives in CS, then pointers can't be left out. If it's about C++, especially good style C++, then pointers should be explained as nearly the last thing. – stefan Jul 29, 2014 at 19:49

Agreed entirely; I got the impression that it's about CS primitives (at least that's what I assume intro to programming would be), but I suppose only the OP could confirm either way. :-]
– ildjarn Jul 29, 2014 at 19:55

The course is twofold. From one side it is "learning to solve problems by devising a solving procedure and implement it as a computer program". This happens to be done in C++11. From the other side it is "learn the basics of C++11 as the basis of any other language". In other words, you should know what a pointer is before moving to Java, so that you can appreciate (or depreciate) garbage collection by reachability. – Ilio Catallo Jul 30, 2014 at 8:33