Is a pointer an lvalue or rvalue?

Asked 10 years, 5 months ago Modified 3 years ago Viewed 6k times

7

4

In other post, I came across

> (5.2.9/8) An rvalue of type "pointer to member of D of type cv1 T" can be converted to an rvalue of type "pointer to member of B of type cv2 T", where B is a base class (clause 10) of D,

Note this from language standard. so my question,

```
int i = 0;
int *p = &i;
*p = 1;
```

Is pointer an lvalue in all the cases? When does it is treated as rvalue?

c++ pointers

Share Edit Follow Flag

asked Dec 9, 2011 at 19:08
user1086635
1,506 • 2 • 14 • 21

2 in the second line of your program, the expression `&i` is an expression of type `int*`, and is thusly an example of a pointer being treated as an rvalue. You're asking taking the result of the expression `&i` and using it as the rvalue to the assignment operator, with `int *p` as the lvalue of said operator. – matthias Dec 9, 2011 at 19:28

2 Answers

Sorted by: Highest score (default)

22

A pointer is not the kind of thing that can be an rvalue or an lvalue. A pointer is a *type*. The only thing that can be an rvalue or an lvalue is an *expression*.

Consider this similar question: "Is an integer an lvalue or an rvalue". Well, neither. "3" is an integer, and an rvalue. "3=i;" is illegal. But "i=3;" is legal if 'i' is an integer. So 'i' is an integer and an lvalue. '3' is an integer and a rvalue.

Share Edit Follow Flag

answered Dec 9, 2011 at 19:16
David Schwartz
173k • 17 • 199 • 267

Can you please use my example to show whats going on? and why does language standard says? thanks. – user1086635 Dec 9, 2011 at 19:41

2 I'm not sure what you're asking exactly. The rules for whether a pointer is an lvalue or rvalue are the same as for any other expression. Just as "3" is an rvalue, "new foo(3)" is an rvalue. Just as "j" is an lvalue if 'j' is a variable of type integer, so "j" is an lvalue if 'j' is a variable of type pointer. Just as a const reference of type integer is an rvalue, so is a const reference of type pointer to integer. The type is just irrelevant. – David Schwartz Dec 9, 2011 at 20:21

As understood, you wrote: A pointer cannot be an lvalue or rvalue. So in my example: is 'p' not an lvalue? because I can take its address as &p. – user1086635 Dec 9, 2011 at 20:32

3 Yes, the *expression* `p` is an lvalue. And it is also of pointer type. Your question is like "Does an idea have four letters?" When I say "ideas are not the kinds of things that can have four letters", you reply "But *idea* does have four letters!". An *expression* can be an lvalue or an rvalue. A pointer cannot. An expression of pointer type can be an lvalue or an rvalue, just as a *word* can have four letters, whether it's the word "idea" or not. – David Schwartz Dec 9, 2011 at 20:41

I think as @DavidSchwartz pointed out, one should realize that lvalue and rvalue is property of expression. In expression x = 5; x is lvalue and 5 is rvalue, consider another expression x = y here x is l-value and y is rvalue. (Please someone correct me if I make mistake here-> int x; is not a expression, its more of a statement. The property of lvalue and rvalue only come into life as property of expression. – pokche Aug 10, 2016 at 6:21

1

According to c standard 2011:

> Except when it is the operand of the sizeof operator, the _Alignof operator, the unary & operator, the ++ operator, the -- operator, or the left operand of the . operator or an assignment operator, an lvalue that does not have array type is converted to the value stored in the designated object (and is no longer an lvalue); this is called lvalue conversion.
>
> ...
>
> The name ''lvalue'' comes originally from the assignment expression E1 = E2, in which the left operand E1 is required to be a (modifiable) lvalue. It is perhaps better considered as representing an object ''locator value''.

therefore, an expression consisting of a modifiable pointer variable can definitely act as lvalue, so is pointer to pointer and so on:

```
int i = 0;

int *p;
p = &i;        // 'p' is lvalue
int *q = p;    // 'p' is rvalue, lvalue conversion

*p = i;        // '*p' is lvalue
i = *p;        // '*p' is rvalue, lvalue conversion

int **pp;
pp = &p;       // 'pp' is lvalue
int **qq = pp; // 'pp' is rvalue, lvalue conversion

int ***ppp;
ppp = &pp;     // 'ppp' is lvalue
int ***qqq = ppp; // 'ppp' is rvalue, lvalue conversion
```

Share Edit Follow Flag

edited Jun 20, 2020 at 9:12
Community Bot
1 • 1

answered Apr 25, 2019 at 15:56
mzoz
1,117 • 1 • 12 • 23