

comp.lang.c: Re: if(a);

## Re: if(a);

**Source:** [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2004-05/1052.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2004-05/1052.html)

---

**From:** Jack Klein ([jackklein\\_at\\_spamcop.net](mailto:jackklein_at_spamcop.net))

**Date:** 05/11/04

Date: Mon, 10 May 2004 21:58:35 -0500

On Mon, 10 May 2004 19:54:26 -0400 (EDT), "Arthur J. O'Dwyer" <[ajo@nospam.andrew.cmu.edu](mailto:ajo@nospam.andrew.cmu.edu)> wrote in comp.lang.c:

>  
> [NB: xpost to comp.std.c added]  
>  
> On Mon, 10 May 2004, Mark McIntyre wrote:  
> >  
> > Martin Johansen wrote:  
> > > if(a);  
> > >  
> > > In this code, to what type does the if statement cast the variable  
> > > "a" on comparison?  
> >  
> > a isn't cast to any type. A cast is when the programmer deliberately  
> > types extra stuff eg  
> > (sometype) foo;  
> > casts foo to a sometype object.  
>  
> For the OP's benefit: The correct term for something that "acts  
> like a cast" without the explicit "(sometype)" is an "implicit  
> conversion."  
>  
> > Back to your original question: 'a' has to be an integral type, or  
> > something that can be converted to an integral type.  
>  
> As Ben said, not quite. 'a' has to be a type which can be  
> validly compared for equality with 0. That is, the line  
> if (a) ;  
> is exactly equivalent to the line  
> if ((a) == 0) ;  
> no matter what the type of 'a' is. If 'a' doesn't have the right  
> type for that second line to make sense, then the first line is  
> invalid as well.  
>  
>  
> Crosspost to c.s.c added because of the following...  
>

Re: if(a);

comp.lang.c: Re: if(a);

> *Question for the experts: Does the C standard clarify what it*  
> *means by "compare equal to 0" anywhere? Is that 0 the literal 0*  
> *of type 'int', or the value you get by converting 0 to the type*  
> *of 'a'? Is there any valid C program that could tell the difference?*  
>  
> -Arthur

What difference does it make? 0 is a valid value for any scalar type.

If you specifically code:

```
if (a == 0)
```

...where a is any scalar type, the following will occur.

The type of the octal signed int literal 0 will be compared to the type of 'a'. Depending on the type of 'a', one of these things will then happen in the virtual machine:

1. 'a' has type signed int, in which case no conversions are performed.
2. 'a' has type unsigned int, in which case 0 is converted to unsigned int.
3. 'a' has an integer type of lesser rank than signed int. If 'a' is an unsigned type, and the entire range of values of that unsigned type cannot be represented in a signed int, 'a' and 0 are both promoted to unsigned int. (think 16 bit implementation where USHRT\_MAX > INT\_MAX). Otherwise 0 is left alone and 'a' is promoted to signed int.
4. 'a' has an integer type of greater rank than int. 0 is converted to the signed or unsigned integer type of 'a'.
5. 'a' has a floating point type (or, I suppose, under C99, a complex type). 0 is converted to 0.0F, 0.0, 0.0L, or whatever the syntax might happen to be for a C99 complex literal (if there are such things, I haven't used this feature).
6. 'a' is a pointer type, in which case 0 is converted to a null pointer of the corresponding type.

With the exception of #6, these are lumped together in the C99 standard in 6.3.1.8 "Usual arithmetic conversions".

Note that in cases 1 through 5, the value of the signed int literal 0 is still 0 after undergoing conversion, if any.

In actuality, I think virtually any compiler in almost every circumstance will use the "as-if" rule to avoid the conversion and use

Re: if(a);

comp.lang.c: Re: if(a);

the most efficient operation available for the underlying architecture to directly test the value of 'a'. Most processors provide special hardware to make a zero/not zero test simple and fast. But under the "as-if" rule, it makes no difference.

Zero is a rather unique concept in mathematics, and it has a well-defined meaning for every scalar type in C. The implementation is free to use whatever method it deems appropriate so long as it correctly determines whether or not the scalar being tested is exactly 0 (or NULL) or not.

--

Jack Klein

Home: <http://JK-Technology.Com>

FAQs for

comp.lang.c <http://www.eskimo.com/~scs/C-faq/top.html>

comp.lang.c++ <http://www.parashift.com/c++-faq-lite/>

alt.comp.lang.learn.c-c++

<http://www.contrib.andrew.cmu.edu/~ajo/docs/FAQ-ac11c.html>