

Re: !!, what is it?

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2005-09/msg01272.html

- *From:* Michael Mair <Michael.Mair@xxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 14 Sep 2005 22:42:49 +0200
-

Tim Woodall wrote:

On Tue, 13 Sep 2005 09:12:42 +0200,
Michael Mair <Michael.Mair@xxxxxxxxxxxxxxxxxxx> wrote:

Tim Woodall wrote:

On Mon, 12 Sep 2005 09:36:01 +0200,
Michael Mair <Michael.Mair@xxxxxxxxxxxxxxxxxxx> wrote:

As an aside:

I recall seeing some platform specific headers which went for the all-bits-one representation of "true" -- but the C implementations gave 1 for !!TRUE as well...

```
int is_it_seven(int x)
{
    return x==7;
}

if(is_it_seven(7) == TRUE)
    printf("7 is seven\n");
else
    printf("7 is not seven\n");
```

While I would never write the explicit test for TRUE [1], I would be horrified at any header that defined TRUE such that this code didn't behave as expected.

[1] Other peoples coding standards excepted.

Re: !!, what is it?

Sorry, without C99's `_Bool` and `_True`, this argument is bogus. Apart from the possible range of return values one could expect from `is_it_seven()` and the possible mismatch with, say `isalnum() == TRUE` or `strcmp() != TRUE`, there is no benefit in that.

I KNOW there is no benefit in that. That's why I wrote "While I would never write the explicit test for TRUE"

But I would not pass review any C source that defined `FALSE` as anything other than 0 and `TRUE` as anything other than 1 (or some equivalent expression)

Infact, assuming I spotted it, I wouldn't accept any code that had a function commented `/* returns TRUE or FALSE */` unless the function returned only 1 or 0 regardless of whether the macros `TRUE` and `FALSE` were actually defined

We certainly agree on that and I did not mean to imply that you did not know that there is no benefit.

However, this does not change the fact that "true" in C prior to C99 is everything which is not zero. To underline this, I mentioned this particular definition of true.

IIRC, this "TRUE" was intended for bitwise operations and conveniently fulfilled `!TRUE == FALSE`. However, this would not have worked on a 1s complement platform.

I still wouldn't have accepted it. There will have been a better name for the macro.

Definitely.

Knowing a better way does not help all the time. I often enough come across the mess of "somebody whose successor left the company some years ago" or similar without the opportunity to fix it (ROI). So, sometimes you have to live with the way things have to be done and do the best to wrap the ugliness and do it better.

Sorry, I probably did not phrase it carefully enough.

Re: !!, what is it?

This was not my question. I am well aware of s-m and ls complement, my question more or less is whether it is possible that a signed int bitfield could be treated as if it was an unsigned int bitfield?

No. (IMO) 6.7.2.1

9 A bit-field is interpreted as a signed or unsigned integer type ...

and then footnote 104) paraphrased - if the type specifier is int it is implementation defined whether the bitfield is signed or unsigned

Which implies to me that signed int -> signed, unsigned int -> unsigned and int goes to one or the other.

Thank you for your opinion on that. This is my reading too. Unfortunately, I have been wrong often enough concerning the standard.

Cheers

Michael

--

E-Mail: Mine is an /at/ gmx /dot/ de address.

.