

Re: mutually referential (Pg 140 K&R2)

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2005-05/msg00110.html

- *From:* Keith Thompson <kst-u@xxxxxxx>
 - *Date:* Mon, 02 May 2005 18:36:11 GMT
-

pete <pfiland@xxxxxxxxxxxxxxxx> writes:

> Dave Thompson wrote:

[...]

>> Forward declared struct and union tags are incomplete, until (the end
>> of) the full definition (if any). You can't forward declare an enum
>> tag; technically an enum type is incomplete from its type-specifier to
>> its closing brace, but this only means you can't use sizeof(enum foo)
>> as one of its own values which seems pretty silly anyway.

>

> I think you can. The enum type is complete,
> that is to say that it's type size is known at compile time,
> even if it's values haven't been specified.

>

> /* BEGIN new.c */

>

> #include <stdio.h>

>

> int main(void)

> {

> enum foo {Zero = sizeof (enum foo), One};

> enum bar;

>

> printf("sizeof (enum bar) is %lu bytes\n",

> (long unsigned)sizeof(enum bar));

> return One - Zero - 1;

> }

>

> /* END new.c */

No. C99 6.7.2.2p4 says:

Each enumerated type shall be compatible with char, a signed integer type, or an unsigned integer type. The choice of type is implementation-defined, but shall be capable of representing the values of all the members of the enumeration. The enumerated type is incomplete until after the } that terminates the list of enumerator declarations.

A footnote says:

Re: mutually referential (Pg 140 K&R2)

An implementation may delay the choice of which integer type until all enumeration constants have been seen.

If your compiler accepts the above code (in conforming mode), it's probably broken. <OT>gcc doesn't.</OT>

—

Keith Thompson (The_Other_Keith) kst-u@xxxxxxx <<http://www.ghoti.net/~kst>>
San Diego Supercomputer Center <*> <<http://users.sdsc.edu/~kst>>
We must do something. This is something. Therefore, we must do this.

.

-
- *Follow-Ups:*
 - ◆ *Re: mutually referential (Pg 140 K&R2)*
 - ◇ *From: pete*
 - *References*