

Re: Forward declarations

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.cpp/2004-05/1350.html

From: Leor Zolman (*leor_at_bdsoft.com*)

Date: 05/11/04

Date: Tue, 11 May 2004 15:16:33 GMT

On Tue, 11 May 2004 16:35:12 +0200, "Marcin Kalicinski"
<kalita@poczta.onet.pl> wrote:

>Hi All,
>
>In some headers meant to work with both C and C++ the following is often
>found:
>
>typedef struct tagMyStruct { /*some members*/ } MyStruct;
>
>Can I forward-declare MyStruct somehow?

Sure, you can say:

```
typedef struct tagMyStruct MyStruct;
```

and complete the definition later on, if necessary.

>
>Or more generally, can I forward declare a typedef or an enum?

typedefs, I don't think so. Under Comeau, in C mode, if I did something like this:

```
typedef b;  
it took it to mean as if I'd said:  
typedef int b;  
(and emitted a suitable warning).
```

With enums, Comeau tells me that forward-declaring them is nonstandard. So it may be supported, but isn't portable.

> Also, is
>there any reason why there should be difference between forward declarations
>of classes and structs (i.e. why I need to explicitly write keyword 'struct'
>or 'class' when forward declaring).

Well, you began this post by saying "In some headers meant to work with both C and C++", so there's an obvious reason to prefer struct right there.

comp.lang.c++. Re: Forward declarations

> *In some cases I don't immediately know*
> *what to write, because, for example, the type definition in original header*
> *file is generated with some macro, such as this:*
>
> *STDINTERFACE(IBlahBlah, IBaseBlahBlah) {*
> */* Members */*
> *}*
>
> *Now I need to search for the definition of STDINTERFACE to see how it works.*
> *Even if I find it, it still may silently change in some future version of*
> *the header file, and cause my compilation to fail. Wouldn't allowing a*
> *generic form of forward declaration, without introducing a new keyword:*
>
> *typename IBlahBlah;*
>
> *be beneficial to C++?*

I would hope that any such header file gives you a documented way to /use/ the facilities it generates, so that such use is immune to future implementation changes. If you think you're using something that's going to morph out from under you in the next release, you'll have to get creative with, perhaps, some preprocessor (or typedef) hackery of your own. Don't hold your breath waiting for changes in the language to support you here.

–leor

>
> *best regards,*
> *Marcin*
>
>
>

--

Leor Zolman --- BD Software --- www.bdsoft.com
On-Site Training in C/C++, Java, Perl and Unix
C++ users: download BD Software's free STL Error Message Decryptor at:
www.bdsoft.com/tools/stlfile.html