

## Re: Newbie question

---

*Source:* [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2005-12/msg02246.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2005-12/msg02246.html)

---

- *From:* Barry Schwarz <schwarzb@xxxxxx>
  - *Date:* Sat, 17 Dec 2005 20:47:10 -0800
- 

On Sun, 18 Dec 2005 04:18:05 -0000, Longfellow <not@xxxxxxxxxxxxxx> wrote:

>After further review:

>

>On 2005-12-17, Flash Gordon <spam@xxxxxxxxxxxxxxxxxxxxxx> wrote:

>> Longfellow wrote:

>

><snip>

>>> What I've done is declare a struct in a header file, which is included

>>> in the files for both ends. For example:

>>

>> Declaring structs, typedefs, function prototypes etc yes, defining

>> objects and function no.

>>

>>> -----

>>> /\* ---- data.h ---- \*/

>>> struct data {

>>> char question[80];

>>> char answer[80];

>>> } calc[80];

>>> -----

>>

>> Here you are not merely defining the struct, you are also defining an

>> object called calc which is an array of structs.

>

>Not defining, just initializing. This declaration is made only once in

>the header file and not again referenced in the code. It is assumed

>that every file that has that header included should know what it needs

>to know about that array of structs.

No, this is a definition. If you include your header before any functions, the array calc will be defined with external linkage. If the header is included in two different translation units, your linker will complain about duplicates.

>

>>> This works fine. My question is this: Is this legal and conforming?

>>

## Re: Newbie question

>> No, absolutely not. Any object that you use must be defined EXACTLY  
>> once, and because data.h is included in more than one source file any  
>> objects it defines are defined multiple times.

>

>Prohibition is against declaring an object more than once, I think. H&S  
>says on page 79 that one of the two exceptions to this rule is when an  
>object is declared as the same type every time, as in a header file that  
>can be shared with all code files. You are correct that an inclusion of  
>the header file in each code file is a duplicate declaration, but this  
>exception exists because otherwise the standard C library would be  
>illegal.

>

>> You want something like:

```
>> /* --- data.h --- */
```

```
>> #define CALC_SIZE 80
```

```
>>
```

```
>> struct data {
```

```
>> char question[80];
```

```
>> char answer[80];
```

```
>> };
```

```
>> extern struct data calc[CALC_SIZE];
```

```
>
```

>This does not compile.

>

>> Then, in EXACTLY one C source file (not header) which includes data.h

>> you have

```
>> struct data calc[CALC_SIZE];
```

```
>
```

>See above.

>

>>> I get from K&R2, page 82, that functions can indeed be declared in

>>> header files. Do I understand this correctly?

>>

>> Yes, but only if you understand the difference between a declaration  
>> (which says basically this thing exists somewhere, but I'm not defining  
>> it here) and a definition (which says OK, this is the actual  
>> object/function right here so allocate the space/put the code here).

>

>Apparently an initialized declaration at the top level is considered a  
>legal defining declaration [H&S pp113–116].

>

>For my purposes, having this array of structs globally available via a  
>header file serves my purpose. Where I don't need it, the header file  
>will not be included, and I'm going to make the variable names unique  
>enough so that they will not be duplicated elsewhere.

>

>In any case, it compiles cleanly with `-Wall -ansi -pedantic`. I've no  
>idea how this setup will fare as I flesh it all out, but I think I'll  
>drag out lclint and use it continuously. That way, I'll know that I'm  
>staying out of trouble.

>

Re: Newbie question

>Thanks anyway,  
>  
>Longfellow

<<Remove the del for email>>

.

---

• **References:**

- ◆ **Newbie question**
    - ◇ *From:* Longfellow
  - ◆ **Re: Newbie question**
    - ◇ *From:* Flash Gordon
  - ◆ **Re: Newbie question**
    - ◇ *From:* Longfellow
- 
- Prev by Date: ***what data structrue book do you use or recommend?***
  - Next by Date: ***Good C materials***
  - Previous by thread: ***Re: Newbie question***
  - Next by thread: ***Re: Newbie question***
  - Index(es):
    - ◆ ***Date***
    - ◆ ***Thread***