

## Re: Simple question about headers and malloc!

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2007-12/msg00979.html>

---

- *From:* "David Webber" <dave@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
  - *Date:* Thu, 27 Dec 2007 12:31:42 -0000
- 

"Robby" <Robby@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message  
[news:81F60840-6CBB-4815-9D2E-EE2DB021C218@xxxxxxxxxxxxxxxxxxxx](mailto:news:81F60840-6CBB-4815-9D2E-EE2DB021C218@xxxxxxxxxxxxxxxxxxxx)

I find myself in a bind where I *must* specifically include the yyy header file (see code sample below) in main even though all its declarations are specifically used for yyy.c. Therefore I am making all of its declarations global to the whole program! and I don't like it !

The whole point of declaring function prototypes in yyy.h of functions in yyy.c is that you can include yyy.h also in the files which contain calls to the functions in yyy.c, and the calling functions know what they're calling.

It isn't global to the whole program unless all .c files used to build the program have calls to things in yyy.h. In which case the function prototypes *must* be available to all of them.

The reason for this is because when I am in yyy.c file I am allocating memory (using malloc) and then exit back to main. I later call again functions in yyy.c and allocate another block of memory (using malloc) and further continue some logic in yyy.c. As I read data from first memory allocation, I get data stored from the second allocation... even though I didn't free the first malloc allocation! I don't free it because I need to use both data in the memory allocated by both mallocs. But each allocation is done at different times!!!!

Therefore, I think that when you exit a function where you used malloc, its pointer is lost right? ...

Only if you lose it, and you mustn't.

If you allocate memory, it is your responsibility to keep track of a pointer to that memory, and free the memory when you have finished with it.

malloc will not normally allocate memory overlapping with already-allocated memory. You must be doing something horribly wrong.

## Re: Simple question about headers and malloc!

At a later instance if I go back in yyy.c and do another malloc , since the pointer to the previous memory allocation is lost, I think that the previous memory block is up for grabs by the next malloc command

No. You have allocated memory and forgotten where it is. You haven't freed it. You have what is known as a memory leak.

and this is where data gets mixed up. So this is why I am including the yyy.h file in main so all pointers returned from mallocs always stay valid and all allocations are reserved at all times.

That shouldn't help. But have you got something horrible in yyy.h???

There are other files between xxx.c and yyy.c and would be quite lengthy to explain it all,

Then there is a lot of scope for something unexplained to go wrong.

but I hope its clear enough to get a general idea.

A "general" idea is usually insufficient for debugging.

Making all pointer declarations that will be used with malloc commands in a global header file

Ah, now you're talking. You have just declared a pointer in the header file – eg

```
int * pMyInteger;
```

?????

And you're using it in two .c files? This is not a good idea. Have you stepped through the program to see where it is pointing in the two different .c files? One .c file will have a pointer called pMyInteger and so will the other one. They will not be the same.

There are ways to do this: for example put `int *pMyInteger;` in one of the .c files and declare

```
extern int *pMyInteger;
```

Re: Simple question about headers and malloc!

## Re: Simple question about headers and malloc!

in the other. But this use of global variables is pretty horrible style in almost all cases.

...is the only way I see that preserves memory allocations when mallocs are done at different times since the pointers returned by malloc are never lost.

It won't.

I don't really think I would like to return pointers returned by malloc back to main in order to preserve them.

Well, I'm afraid you should reorientate your "likes" a bit, for that is the way it is done. You allocate memory, and you hang onto the pointer (not another one in another file with the same name) until you free the memory. Passing it back to the function which "owns it" is the way this is done.

```
void MyFuncion()
{
int *pMyInt = someOtherFunctionWhichAllocatesMemory( ... );
//.....
someOtherFunctionWhichFreesMemory( pMyInt );
pMyInt=NULL; // For safety as it is no longer valid.
//....
}
```

The called functions can be malloc and free, or they can be your own functions which call malloc and free somewhere down the line.

...  
This obviously compiles

Few things are obvious in this life, and this is not one of them. :-(

but  
not sure if this is a good C programming practice. Any feedback appreciated.

```
#include <yyy.h> //Is this okay?
#include <xxx.c>
```

Have you really included xxx.c in another .c file?

Re: Simple question about headers and malloc!

```
int main()
{
//...some code !
//Calls functions in xxx.c
}
```

```
=====xxx.h
...some declarations
=====
=====xxx.c
#include <xxx.h>
#include <yyy.c>
```

Have you really included yyy.c in another .c file?

```
...some code
//Calls functions in yyy.c
=====

=====yyy.h
...some declarations
=====

=====yyy.c
//Should yyy.h be included here instead?
...some code
=====
```

I think you need to go back to an introductory text on c and study the differences between header files and source code files, what #include means, and what "linking" means. I'm afraid the above is hopelessly confused. :-(

Dave

--

David Webber  
Author of 'Mozart the Music Processor'  
<http://www.mozart.co.uk>  
For discussion/support see  
<http://www.mozart.co.uk/mozartists/maillinglist.htm>

.