

Re: #define ALLOCIT(Type) ((Type\*) malloc (sizeof (Type)))

# Re: #define ALLOCIT(Type) ((Type\*) malloc (sizeof (Type)))

---

*Source:* [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2007-02/msg01465.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2007-02/msg01465.html)

---

- *From:* "Nick Keighley" <[nick\\_keighley\\_nospam@xxxxxxxxxxx](mailto:nick_keighley_nospam@xxxxxxxxxxx)>
  - *Date:* 9 Feb 2007 07:53:00 -0800
- 

On 9 Feb, 14:52, Yevgen Muntyan <[muntyan.removet...@xxxxxxxxx](mailto:muntyan.removet...@xxxxxxxxx)> wrote:

Consider the following macro:

```
#define ALLOCIT(Type) ((Type*) malloc (sizeof (Type)))
```

well several million points for courage...

The intent is to wrap raw memory allocation of N bytes into a macro which returns allocated chunk of memory to be used as a Type structure.

The background: I was beaten many times (it surely is not my exclusive experience) by functions which return and accept void\*.

could you give some examples? I can't see how a function returning void\* can "beat you". Assign to the wrong type?

There are such functions, and there are cases when you can't do much about it (third-party libraries, containers, etc.). I make a conclusion: void\* is evil and must be avoided when possible and easy.

I don't use it heavily, but I wouldn't catagorise it as "evil". malloc() seems ok to me.

Therefore I believe it's better to use above macro than raw malloc. And I think this is a valid belief, not like believing in Santa or alive

Re: #define ALLOCIT(Type) ((Type\*) malloc (sizeof (Type)))

Re: #define ALLOCIT(Type) ((Type\*) malloc (sizeof (Type)))

Elvis.

well you presumably have some evidence of lowered bug rates or ease of use or whatever for your macro.

In toy programs like posted here, when you do

```
int *func (void)
{
int *foo = malloc (10 * sizeof *foo);
...
}
```

it certainly doesn't matter how you use malloc since it's small and easy. But in big programs, when you have lot of other things to worry about, it's nice to know that you reduced number of function calls which return void\*. At least it's nice to know for me. Type mismatch here is quite a usual thing (unfortunately), since I deal a lot with "classes", with nested structures like

```
struct Parent {...};
struct Child {struct Parent parent; ...};
```

It's extremely easy to mess it up, and avoiding using extra void\* is actually a good thing.

I don't really see why. In "clc" idiom:

```
Parent *p = malloc (sizeof *p);
```

In your idiom

```
Parent *p = ALLOCIT (Parent);
```

Since your version has the type in two places I'd say an error was \*more\* likely. If my example is misleading could you construct a better one?

I don't see how your macro improves type safety.

Note that this

Child-Parent thing has nothing to do with malloc.

Re: #define ALLOCIT(Type) ((Type\*) malloc (sizeof (Type)))

Re: #define ALLOCIT(Type) ((Type\*) malloc (sizeof (Type)))

this confuses me. I thought this discussion was about malloc()?

But  
it does have a lot to do with using wrong types. And  
I don't try to think like "this is malloc, has nothing  
to do with that, void\* is fine here".

I don't understand this bit.

I just avoid using void\*.

Now not all people think that ALLOCIT() buys any type safety.  
Some other people do think it does.

Anyway, it's hard to argue with more than one person  
at once, so I decided to post what I think about this  
particular "yes I want cast \*here\*" thing. You may  
say I want C++, you may say "Wrong.", you may prove  
using ten steps that it's wrong, whatever. I at least  
know that I am not the only person who believes in  
"cast is good in this single thing" so I am not a complete  
idiot.

I might be better to come up with a reasoned technical  
argument.

(Of course you can say that people who believe this  
are all idiots or wrong, but working software proves they  
aren't).

well not necessarily. The cost in \$/function–point may higher.  
I'm not saying it is, but simply producing "working" software  
doesn't tell you everything.

--

Nick Keighley

– Note:  
– I wanted to put a check in getGuiExtract(), but it's a void  
method, inherited from the father of the father of the father of  
the  
father of the father of the father of the father of the

Re: #define ALLOCIT(Type) ((Type\*) malloc (sizeof (Type)))

Re: #define ALLOCIT(Type) ((Type\*) malloc (sizeof (Type)))

father...

So I can't modify its interface.  
(MS found in C++ comment)

.