

Re: Is it better to use a macro or a function?

Re: Is it better to use a macro or a function?

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2007-06/msg01422.html

- *From:* Johan Bengtsson <qwerty_42@xxxxxxxxxxx>
 - *Date:* Sat, 09 Jun 2007 10:39:52 GMT
-

Ben Bacarisse wrote:

Johan Bengtsson <qwerty_42@xxxxxxxxxxx> writes:

Ben Pfaff wrote:

madhawi <madhawi.k@xxxxxxxxxxx> writes:

Is it better to use a macro or a function?

Some tasks can only be accomplished with a macro.
Otherwise, use
a function: in general, they're safer.

An example:

I have in some of my programs a lot of pointers to structs.

```
struct a
{
char *s1;
char *s2
} *b;
```

```
char *s;
```

A general problem then is that such a pointer might be NULL and therefore referencing a member of that struct is illegal.

```
s=b->s1; /*if b is NULL this doesn't work well...*/
```

Now if what I want is something like this:

```
if (b)
s=b->s1;
else
s=NULL;
```

That is I want to assign **something** to s, if b is NULL (or for that

Re: Is it better to use a macro or a function?

matter `b->s1` is NULL) I do want `s` to be NULL too. This could be written:

```
s=b?b->s1:NULL;
```

That's OK, so now if `b` isn't really a variable but rather a long complex expression I need to type that twice to get the desired effect.

```
#define spt(ptr)(!(ptr))?NULL:ptr
```

would make it possible to write

```
s=spt(b)->s1;
```

That trick is *not* possible with a function!

Did you consider:

```
struct a *spt(struct a *b)
{
    static struct a null_a = { NULL, NULL };
    return b ? b : &null_a;
}
```

Sure, you need one for each structure type, that is one thing macros are *really* good for. You get more type safety, and when the time comes (as it always does on my programs) you have somewhere to put your `printf`s!

I might be dumb, but I don't see how that solves the same thing as the macro.

I repeat myself...

```
#define spt(ptr) (!(ptr))?NULL:ptr
a=spt(b)->c;
expands to
a=(!(ptr))?NULL:ptr->c;
```

now if `ptr` is not NULL: (normal case)

```
a=ptr->c;
and if ptr is NULL:
a=NULL;
```

You see? the `->c` part "disappears" because of the macro even if it is outside what looks like the call, and that is the code that might actually crash the program.

And the part about type safety, well I would probably need hundreds (if not thousands) of those functions in that project... (I could write it so I have one function for each struct *and* each member of that struct, but that would be a *lot* of functions)

Re: Is it better to use a macro or a function?

Re: Is it better to use a macro or a function?

And the speed would suffer too I think... I guess (but have not verified) that the speed difference (on most systems with fairly good optimizing compilers) with the macro (vs not having any checks at all) would be that of the two assembler instructions needed to compare the pointer to NULL and then make a conditional jump.