

Re: Macros

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2004-02/0091.html>

From: Louis Solomon [SteelBytes] (louis_at_steelbytes.spam-is-bad.com)

Date: 02/03/04

Date: Tue, 3 Feb 2004 13:14:18 +1100

life is not simple.
and neither are all the subtiles of optimizing code.

some rules of thumb

- * stack variables vs static/global vars are not very different in speed or size.
- * member variables vs stack & static/global vars are slower
- * virtual methods are usually slower then non virtual.
- * non virtual methods are similar/same to static/global funcs for speed.
- * only unroll 'inner' loops (and likewise, only inline 'inner' funcs).
- * there is nothing like profiling to find out which is quicker. (run code using both methods 1000 times in a loop, and compare times).

--

Louis Solomon

www.steelbytes.com

"Bonj" <a@b.com> wrote in message

news:u%23PYMd6DHA.2404@TK2MSFTNGP11.phx.gbl...

> Consider the two following possible implementations of a program:

>

> one, written in C++. With classes (let's say, 2, of which there is
> probably

> only about 5 instances instantiated at any one time). Lots of local
> variables and stack usage. Class isn't padded with dummy variables,
> although

> byte alignment is set to 16 bytes (the maximum). The class has mainly 4
> byte

> variables although one is a 16 byte one (a RECT).

>

> the second, written in plain old C. It is the same program as the first,
> and

> in a lot of places is pretty much similar code. But instead of having a
> class (as C can't), it has a struct, all variables being 4 bytes (there is
> no RECT in the struct). The struct is padded manually to 128 bytes.

> Instead

> of class functions, there are macros which operate on the struct. Instead
> of

> having local variables, it has globals - so there is no usage of the stack
> so to speak for each function - it just does a job. Every function that
> returns a void (about 80% of them) is then converted into a macro, so
> that

> the only functions that the program actually needs to generate function
> overhead for are the system functions, such as WinMain and the WndProc,
> and

Re: Macros

microsoft.public.vc.language: Re: Macros

> a couple of other main controlling functions - the rest is totally inline.
>
> Understandably, the first will obviously be a lot neater code. But I've
> decided to consider investigating the possibility of the second - to try
> to
> gain as near to machine code performance as possible (even though I can't
> begin to understand machine code). So far it seems alright, and not too
> ugly
> (to me, anyway).
> ASIDE from the benefits of having cleaner code - just speaking in terms of
> performance, how much better is the second approach likely to be? In the
> region of 0.000001% ? or maybe 2% - 10% ? Or maybe about 0.1% - 1%?
>
> Those are the options.
>
> Comments?
>
>