

Re: VS2008 destroys static objects before global (non-static) objects?

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2008-10/msg00637.html>

- *From:* Boris <boriss@xxxxxx>
 - *Date:* Wed, 22 Oct 2008 15:07:29 +0200
-

On Wed, 22 Oct 2008 13:28:30 +0200, Ulrich Eckhardt <eckhardt@xxxxxxxxxxxxxxxx> wrote:

Boris wrote:

Can anyone confirm that the function `_CRT_INIT` has been changed from VS2005 to VS2008 in such a way that static objects are now destroyed first before global (non-static) objects are destroyed? Looking at the latest C++ standard draft this seems to be permitted (and would explain why I see a crash when closing a software which I just upgraded from VS2005 to VS2008).

I can't confirm nor deny that particular change, but the initialisation order of globals in different translation units has always been unspecified, hence the so-called "initialisation order fiasco". Other than that, objects are generally destroyed in the opposite order of their creation and at least function-static objects are therefore always destroyed before globals. What kind of static are you referring to?

My problem explained in pseudocode is this:

```
-----  
template <typename T>  
class memory_manager  
{  
public:  
void *operator new(std::size_t) { return mem_p_.malloc(); }  
void operator delete(void *p) { p_.free(p); }  
  
private:  
static pool p_;  
};  
  
class small_object : public memory_manager<small_object>  
{  
};
```

Re: VS2008 destroys static objects before global (non-static) objects?

```
std::auto_ptr<small_object> o(new small_object());  
-----
```

While the class definitions are in one DLL `o` is defined in another DLL (let's call them `1.dll` and `2.dll` where `1.dll` depends on `2.dll`). When the DLLs are loaded `p_` in `2.dll` is initialized first followed by an initialization of `o` and again `p_` in `1.dll` (at least that's what I see when stepping through the DLL initialization code; maybe I'm wrong here as I don't fully understand what Microsoft is doing there in `crt.dll.c` and `crt0dat.c`). When the DLLs are unloaded everything is uninitialized but in reverse order. As the destructor of `p_` is then called first which frees all memory managed by the memory manager I get an access violation when `o` is destroyed.

The code worked fine for years with VS2005 (and also `g++` on Linux). If the initialization and destruction order was really changed in VS2008 I need to think about how to adapt the code. :-/

Boris

.