I'm not calling a virtual function from my constructor or destructor, but I'm still getting a __purecall error

devblogs.microsoft.com/oldnewthing/20120830-00

August 30, 2012


Raymond Chen

Some time ago, I described what `__purecall` is for: It's to detect the cases where you call a virtual method with no implementation (a so-called *pure virtual method*). This can happen during object constructor or destruction, since those are times when you can validly have a partially-implemented object. Well, there's another case this can happen: If the object *has already been destructed*. If you call a method on an object that has already been destructed, the behavior is undefined. In practice, what happens is that the method runs on whatever leftover values are in memory where the object used to be. Depending on how lucky or unlucky you are, this may resemble the actual object closely enough that the method doesn't notice, or at least doesn't notice for a while, or maybe it can't even get off the ground.

The `__purecall` error is a case where it can't even get off the ground. The leftover object still has a vtable, namely the vtable of the base class, the one that has `__purecall` entries for all the pure virtual functions. (It still has that vtable because that's the object's last identity before finally going invalid.) And the method you tried to call is a virtual method that is pure in the base class. Not only are you calling a pure virtual function after destruction has begun, you're calling it *after destruction is complete*.

Raymond Chen

Follow