

Re: How many bytes per Italian character?

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-04/msg00895.html>

- *From:* "Doug Harrison [MVP]" <dsh@xxxxxxxx>
 - *Date:* Mon, 09 Apr 2007 16:01:47 -0500
-

On Mon, 9 Apr 2007 10:39:35 -0700, "David Ching"
<dc@xxxxxxxxxxxxxxxxxxxxxxxx> wrote:

OK, now I see where you're headed with this. The derived class needs to override the `InitState()` function so it needs to be pure virtual. And it needs to call the base class's `InitState()` function somewhere, either in its implementation of `InitState()` or somewhere else, that's why it must not be private. (But pure virtual functions do not force the derived implementation to call the base's implementation, I don't think.)

Neither do normal virtual functions, but many require that overrides in derived classes call up to them anyway; an example is `OnInitDialog`. It's a documentation issue, like I said.

So the real problem is that C++ jams two conflicting functionalities into pure virtual functions. 1) Forcing derived classes to implement it; 2) Providing a convenient place to implement base functionality for access by both the base class and derived class.

These are somewhat contrary goals. The common sense usage of pure virtual functions is to make it pure if the base class has no idea how to implement the functionality, and that must be provided by the derived classes. If the base class does have some idea of it, and therefore needs to implement the virtual function, then don't make it pure.

If the derived classes are developed hand in hand with the base as a unified solution to a problem, and you know the base can't implement the virtual function completely, there's no conflict.

The derived classes have the opportunity to supplement/replace the functionality by overriding the virtual function or not, and if they do override, whether or not to call the base class.

Re: How many bytes per Italian character?

The purpose of the derived class in my example is to add its own stuff to what the base class does. There's no choice here.

It's presumptuous of the base class to assume the derived classes will be implemented in such a way that they are forced to override in this case.

Not at all. FWIW, I did characterize defining pure virtual functions in the terms, "Offhand, the only time I can recall doing this..." IOW, it isn't common IME, but I did appreciate the ability to express what I needed to do in a short, clear way. To the extent it addressed what I was talking about, the alternative you presented requires a parallel set of non-virtual functions to be defined in order to keep your idea of "pure", well, "pure". It doesn't ease the documentation requirement, so it doesn't really gain anything I consider valuable. It's just a harder way to do it.

(To put this in C# terms, I defined an interface, had the base class implement it, and then had the derived classes reimplement it but still call the base class as part of their implementation.)

--

Doug Harrison
Visual C++ MVP

.