Subject: Re: More efficient method of appending to arrays when using pointers?
Posted by Matt Francis on Tue, 04 Jan 2011 22:39:00 GMT
View Forum Message <> Reply to Message

On Jan 5, 9:30 am, Gray <grayliketheco...@gmail.com> wrote:
> On Jan 4, 5:01 pm, Matt Francis <mattjamesfran...@gmail.com> wrote:
>
>
>
>> I have some code I've written that looks clunky and I was wondering if
>> there is a more efficient (faster and or using less memory) way to do
>> this.
>
>> I am using a custom object with a member self.foo which will end up
>> being a matrix, built up by appending arrays one at a time as I loop
>> over each step of a process. This update code currently looks like
>> this:
>
>> temp = [*(self.foo),next_array]
>> ptr_free,self.foo
>> self.free = ptr_new(temp)
>
>> This seems to be a bit wastefull in terms of how many times memory is
>> allocated and deallocated to get the job done. Something simple like
>
>> self.free = ptr_new([*(self.foo),next_array]
>
>> causes a memory leak due to the dangling pointer. I don't see how the
>> TEMPORARY function can be used here without causing a leak.
>
>> Any tips from the pros?
>
> Why mess about with ptr_new and ptr_free? Unnecessary.
>
> temp = [*self.foo,next_array]
> *self.foo = temp
>
> Or, the minimalist approach:
>
> *self.foo = [*self.foo,next_array]

Hmmm, I didn't expect that would work! Still keep forgetting IDL
'pointers' aren't really pointers and can do funny things (I'm a C++
programmer and this wouldn't work with a 'real' pointer).

Thanks for your help.