Subject: Re: Pointer Behavior Objects Vs Plain routines?
Posted by David Fanning on Wed, 11 Sep 2002 15:17:14 GMT
View Forum Message <> Reply to Message

savoie@nsidc.org (savoie@nsidc.org) writes:

> O.k. I'm looking at some pointer weridness. Well, I'm calling it weirdness
> because I obviously don't understand something that is happening. There are
> two examples below.
>
> The first is just two routines. test: creates a pointer, calls changePtr
> with a null pointer as an argument; and changePtr: which just assigns a
> string the the passedPtr. This examples shows that if you pass a pointer to
> a procedure, assign something to that pointer, you can retrieve it after
> exit.
>
>
> The rest of the routines are a simple object with a couple of methods,
> showing exactly the opposite effect. When the object's CHANGEPTR method is
> called, self.myptr doesn't seem to be able to be changed on return.

The problem here has nothing to do with either pointers
or objects. The problem is that structure dereferences
(I.e., self.myptr) are passed by value, whereas passing
the pointer itself (I.e., myptr) is passed by reference.
Procedures can change things that are passed by reference.
They work on *copies* of things that are passed by value.

Since your DOIT method is a self method, you can simple
change it like this:

```
PRO WEIRD::DOIT
  self.myptr =  ptr_new('Why can not I change this?')
END
```

Then, call it like this:

```
   self -> Doit
```

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com