Subject: Re: Preferred way to get multiple returns from a function
Posted by Paul Van Delst[1] on Mon, 14 Feb 2011 15:03:06 GMT
View Forum Message <> Reply to Message

James wrote:
> I am writing a function that fits an ellipse to a 2*N array of
> points. There are three values to return: the center, semi-major
> axis, and semi-minor axis. This is a simple program, but it brings up
> a more general question: what is the preferred IDL way to return
> multiple values from a function?
>
> Currently, my program returns a structure containing the elements
> {center, major, minor}. However, a lot of built-in IDL routines take
> named variable inputs that are set to the appropriate value on output
> - so instead of something like:
>
>   ellipse_struct = fit_ellipse(points)
>
> I would have:
>
>   fit_ellipse, points, center, major, minor
>
> I'm not sure which is better. C programming has taught me to
> appreciate structures, but I'd like to code in the conventions of the
> language. Which would you prefer, and why?

Structure.

Why? Because it produces self-documenting code.

Similar to what R.G.Stockwell said,

```
  ellipse.center
```

doen't require a comment describing what it is. However, a standalone variable

```
  center
```

probably does. What is it the centre of? An ellipse? Circle? Generic ROI?

"Encapsulation" may be a bit of an OO buzzword, but even for procedural languages with structures it's an easy way to
make code more readable and simple to maintain. That may not be an issue for a person or two writing code, but in a
project where there are many people contributing (and in different timezones) it can be extremely helpful.

And since IDL went OO, I think some of the conventional idioms can be tossed - particular those

that are purely for
procedural languages.

FWIW, I'm dealing with the same issue in Fortran. Ever since it went OO with the Fortran2003 standard, I'm writing all
new code with an OO bent. Makes home-grown "toolboxes" much easier to reuse.

cheers,

paulv

p.s. And always always use Mike Galloy's unit testing framework too: http://mgunit.idldev.com/ :o)