Subject: Re: Passing more than one variable out of a function.
Posted by Maarten[1] on Wed, 25 Mar 2009 07:56:59 GMT
View Forum Message <> Reply to Message

On Mar 25, 5:31 am, Ben <Benjamin.R.Ev...@gmail.com> wrote:

> The question is how can I write a user defined function which can pass
> more than one variable out to the calling program. Just like the WHERE
> function?
>
> I know that I could just use a procedure to do this. But I am just
> curious.

Just like a procedure actually.

```
function blah, in, out
   out = 2*in
   return, sqrt(in)
end

r = blah(4, out)

print, r, out
; prints: 2  8
```

Maarten

Subject: Re: Passing more than one variable out of a function.
Posted by philipelson on Wed, 25 Mar 2009 08:14:33 GMT
View Forum Message <> Reply to Message

```
On 25 Mar, 04:31, Ben <Benjamin.R.Ev...@gmail.com> wrote:
> The IDL intrinsic function WHERE can pass more than one variable out
> to the calling program.
>
> For example:
>
>> array=[0,2,4,6,8,10]
>> arr_subscripts = WHERE(array GT 5, count)
>> print, arr_subscripts
>
> 3,4,5
>
>> print, count
>
> 3
>
```

> The question is how can I write a user defined function which can pass
> more than one variable out to the calling program. Just like the WHERE
> function?
>
> I know that I could just use a procedure to do this. But I am just
> curious.

Hi Ben,

WHERE is simply passing an array back to you, you can do the same with the example code below:

```
FUNCTION test, count=count
  IF ARG_PRESENT(count) THEN count=3
  RETURN, [3,4,5]
END

print, test(count=mycountvar)
print, mycountvar
```

The important thing to notice is that RETURN can accept an array as a parameter.

Hope this helps.

Cheers,

Philip

Subject: Re: Passing more than one variable out of a function.
Posted by Mike[2] on Thu, 26 Mar 2009 13:12:45 GMT
View Forum Message <> Reply to Message

On Mar 25, 4:14 am, philipel...@googlemail.com wrote:

> The important thing to notice is that RETURN can accept an
> array as a parameter.

RETURNing anonymous structures is a handy way to pass back results with mixed types. As in:

```
function do_this, x
result = {timestamp: systime(), $
        mean: mean(x), $
        min: min(x), $
        value:some_other_calculation_using_x(x) $
```

```
      }
return(result)
end

foo = do_this(x)
print, foo.timestamp
print, foo.mean
print, foo.min
print, foo.value
```

Regards, Mike
