
Subject: Re: Array to Scalar

Posted by [afl](#) on Wed, 23 Aug 1995 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

In article <41fg1d\$69h@mojo.eng.umd.edu>, sanjay@windvane.umd.edu (Sanjay K) writes:

|> I noticed, accidentally, that the multiplication of an array
|> by a scalar and multiplication of an array by an array of length 1
|> gives two different answers. Even though this is to be expected
|> for general arrays, there needs to be an exception for array of
|> size 1.

|>
|> Consider the following example:

|>
|> a=findgen(10)
|> ; unrelated code
|> factor=interpol(...)
|>
|> factor returns array of size 1 and if I use
|>
|> a=a*factor
|>
|> I am left with just one value whereas I expect
|> an array of size 10 each of the elements multiplied by factor!
|>
|> My 2 cents worth!.

Yes. This is the exact behavior one should expect.
Maybe you wish to perform a matrix multiply.

```
IDL> a = findgen(10)
IDL> factor= fltarr(1) + 2.0 ; Assign 2.0 to factor vector
IDL> d = a # factor
IDL> help, d
IDL> print, d
```

Andy Loughie
afl@cdc.noaa.gov

Subject: Re: Array to Scalar

Posted by [hahn](#) on Thu, 24 Aug 1995 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

sanjay@windvane.umd.edu (Sanjay K) wrote:

> I noticed, accidentally, that the multiplication of an array
> by a scalar and multiplication of an array by an array of length 1

> gives two different answers. Even though this is to be expected
> for general arrays, there needs to be an exception for array of
> size 1.

[Example deleted]

> -sanjay

What does IDL when a binary (two sided) operation of arrays of
different length is requested ?

```
a = [ 1, 2, 3, 4 ]
```

```
b = [ 5, 6 ]
```

```
help, a*b
```

```
EXPRESSION INT = ARRAY(2)
```

Obviously IDL starts to multiply the arrays element by element
until the shorter array is exhausted. Then the operation terminates.
Thus, if one array is only one element in size, the result will only
receive one element.

There is a big difference between one array element and a skalar
variable:

- * Using an array element as argument of a procedure call
prevents you to change its value! Thus, you cannot read into
an array element.

- * Using `x(*,5)` in an expression will be interpreted as a matrix
rather than a vector. While the assignment `a = x(*,5)`
REFORMats this expression to a vector!

You can probably find more expamples when digging the manual,
this is what just came into my mind...

Norbert Hahn
