Subject: Re: Strange problem
Posted by James Kuyper Jr. on Mon, 26 Nov 2001 18:03:00 GMT
View Forum Message <> Reply to Message

Andre Kyme wrote:

> Hi everyone,
>
> type: for i=0., 0.8., 0.1 do print,i
> That's bad isn't it?

You can get a better idea of what's going wrong by modifying that as follows:

for i=0., 0.8., 0.1 do print,i,format='(f20.10)'

There's a warning about this in the User's Guide, in the part describing the FOR statement.

> Or what about:
>
> for i=0., 9.6, 0.1 do print,i
> for i=0., 9.7, 0.1 do print,i
> Same last element?
>
> If you stick a "d" after the first 0 - ie. make it double precision:
>
> for i=0.d, 0.8, 0.1 do print,i
>
> - then it seems to be OK, kind of
>
> Or try this:
> a=fltarr(100)
> for i=0., 9.7., 0.1 do a[i*10]=i
> Oh darn, 2.3 is not there!
> Neither is 9.3.
> Using the "d" trick seems to fix it, but why the need when we're only
> using steps of 0.1?
>
> Anybody know what's going on?

Yes. You should never use floating point values as your loop variable; there's only a finite set of numbers that can be represented exactly in floating point notation; all other numbers are stored as the nearest approximation from that set. Most decimal fractions can't be represented exactly. None of the numbers you used except 0.0 can be represented exactly. As a float, 0.1 is actually represented by approximately 0.100000001490, and 0.8 is represented by approximately 0.800000011921.

Therefore, you can't guarantee exactly how many times your loop will execute. Also you get roundoff errors which accumulate every time you increment the counter.

My preferrred solution is as follows:

```
for i=0, 8 do print,i*0.1

for i=0, 97 do a[i] = i*0.1
```

The integer increments are exact, so the number of values is guaranteed to be correct. You only get 1 floating point roundoff error, from the multiplication, so the numbers printed are more accurate.