Subject: Re: writing recursive functions
Posted by William Daffer on Wed, 02 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

"Martin Downing" <m.downing@abdn.ac.uk> writes:

Are you using PVWave? fact.pro (take 1) worked fine in IDL 5.1.1.

William

> Hi all,
>
> My aim is to write a function that (recursively) call itself, that can be
> compiled on the fly as with standard code.
> So - whats the problem, well if you _could_ write a function for factorials
> using recursion as:
>
> ---
> ; fact.pro : take 1
> ; example of recursion (I know there's a factorial function!!)
> function fact , n=n
> res = float(n)
> if res gt 1 then $
> res = res * fact(n=res-1)
> return, res
> end
> ; (I used a keyword as otherwise it compiles thinking fact() is a variable)
> ---
>
> ....This will not compile
> It appears that the compiler needs something equivalent to a declaration
> statement (as in C code), so I tried to just write a dummy empty function
> above the real code.
>
> e.g.:
>
> ---
> ;fact.pro : take2
> ; dummy definition I.e. declaration
> function fact , n
> print, "this should not be called!"
> return, n
> end
>
> ; the real code
> ; example of recursion (I know there's a factorial function!!)
> function fact , n

```
>       res = float(n)
>       if res gt 1 then $
>            res = res * fact(res-1)
>       return,  res
>  end
>  ----------------------------------------------------------
>  However, unless this is forcibly compiled (.compile fact.pro), the
>  auto-compile will stop after the initial definition.
>
>  This leaves two options - permanently compiling the code or calling by
>  another routine:
>
>   ----------------------------------------------------------
>  ; call_fact.pro
>  @fact.pro
>  function call_fact, n
>      return, fact(n)
>  end
>   ----------------------------------------------------------
>
>
>  Any suggestions welcome
>
>  Martin,
>  Aberdeen, Scotland
>
>
>
>
>
>
>
>
```

--
Outside of a dog, a book is man's best friend
Inside of a dog, it's too dark to read.
  Groucho Marx.

Subject: Re: writing recursive functions
Posted by davidf on Wed, 02 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Martin Downing (m.downing@abdn.ac.uk) writes:

> My aim is to write a function that (recursively) call itself, that can be
> compiled on the fly as with standard code.
> So - whats the problem, well if you _could_ write a function for factorials

> using recursion as:
>
> ----------
> ; fact.pro : take 1
> ; example of recursion (I know there's a factorial function!!)
> function fact , n=n
> res = float(n)
> if res gt 1 then $
> res = res * fact(n=res-1)
> return, res
> end
> ; (I used a keyword as otherwise it compiles thinking fact() is a variable)
> ----------
>
> ...This will not compile

How about this, using Forward_Function to indicate
that FACT is a function and not a variable. You could
go back to using n as a positional parameter. :-)

```
function fact , n=n
forward_function fact
  res = float(n)
  if res gt 1 then $
       res = res * fact(n=res-1)
  return, res
end
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: writing recursive functions
Posted by Martin Downing on Thu, 03 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Hi all,

My aim is to write a function that (recursively) call itself, that can be
compiled on the fly as with standard code.

So - whats the problem, well if you _could_ write a function for factorials
using recursion as:

```
 ------------------------------------------------------------
; fact.pro : take 1
; example of recursion (I know there's a factorial function!!)
function fact , n=n
    res = float(n)
    if res gt 1 then $
        res = res * fact(n=res-1)
    return,  res
end
; (I used a keyword as otherwise it compiles thinking fact() is a variable)
 ------------------------------------------------------------
```

...This will not compile
It appears that the compiler needs something equivalent to a declaration
statement (as in C code), so I tried to just write a dummy empty function
above the real code.

e.g.:

```
 ------------------------------------------------------------
;fact.pro : take2
; dummy definition I.e. declaration
function fact , n
   print, "this should not be called!"
   return, n
end

; the real code
; example of recursion (I know there's a factorial function!!)
function fact , n
    res = float(n)
    if res gt 1 then $
        res = res * fact(res-1)
    return,  res
end
------------------------------------------------------------
```

However, unless this is forcibly compiled (.compile fact.pro), the
auto-compile will stop after the initial definition.

This leaves two options - permanently compiling the code or calling by
another routine:

```
 ------------------------------------------------------------
; call_fact.pro
@fact.pro
```

```
function call_fact, n
   return, fact(n)
end
 -------------------------------------------------------------
```

Any suggestions welcome

Martin,
Aberdeen, Scotland
