
Subject: functions as arguments ?

Posted by [Udo Grabowski](#) on Wed, 23 Feb 2000 08:00:00 GMT

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

Hello,

I'm looking for a way to do something like the Fortran construct

```
program test
  call proc (func1)
  call proc (func2)
end
subroutine proc(func)
  call func
end
```

where func1,2 are given external functions defined somewhere.
Functional arguments are seemingly not possible in IDL, but I found
a preliminary and ugly solution with a wrapper class:

```
=====
;-----;
; functions finally to be called alternatively ;
;-----;
function func1, n
  return, n+1
end
function func2, n
  return, n+2
end
;-----;
; wrapper class for function ;
;-----;
pro FUNC__define
  struct = { FUNC, i:0 } ;;; (i is dummy)
  return
end

function FUNC::init
  return, 1
end
;-----;
; UGLY solution: ;
; method which calls the hardwired function ;
;-----;
function FUNC::transform, n
  m = func2(n)
  return,m
end
```

```

;-----;
; function f gets its proc from main program ;
;-----;
function f, n, proc=proc
    n = proc -> transform(n)
    return, n
end
;-----;
; main program calls f with functional argument, ;
; but cannot change it dynamically ;
;-----;
pro call_functions

    func_obj = OBJ_NEW('FUNC')

    n = 1
    s = f(n,proc=func_obj)

    print,'n=',n

; any possibility to do anything like: ???
; func_obj -> SetFunction (func1)
; s = f(n,proc=func_obj)
; print,'n=',n

```

end

;=====

This solution is not flexible because I need another wrapper class if I additionally want to call func1.

I also considered pointers, but all these concepts seem to be applicable to data structures only. Is there a more elegant solution to that problem ?

--

Dr. Udo Grabowski email: udo.grabowski@imk.fzk.de
 Institut f. Meteorologie und Klimaforschung II, Forschungszentrum Karlsruhe
 Postfach 3640, D-76021 Karlsruhe, Germany Tel: (+49) 7247 82-6026
 http://www.fzk.de/imk/imk2/ame/grabowski/ Fax: " -6141