



Física Computacional

Escuela de Física

M.R.Fulla¹

¹Escuela de Física, Universidad Nacional de Colombia Sede
Medellín

marlonfulla@yahoo.com- Oficina:21-408

<https://sites.google.com/view/fiscomunalmed/>

September 5, 2023

Problema robusto → Descomposición → Programas unidad funcionales (procedimientos externos).

Procedimientos externos $\left\{ \begin{array}{l} \textit{Subrutinas} \\ \textit{Funciones} \end{array} \right. \left\{ \begin{array}{l} \textit{Compiladas} \\ \textit{Testeadas} \\ \textit{Depuradas} \end{array} \right.$

Beneficios $\left\{ \begin{array}{l} 1. \textit{Prueba independiente de tareas} \\ 2. \textit{Código reusable} \\ 3. \textit{Evitar efectos colaterales} \end{array} \right.$

Subrutinas

```
CALL subroutine_name ( argument_list )

SUBROUTINE subroutine_name ( argument_list )
    ...
    (Declaration section)
    ...
    (Execution section)
    ...
RETURN
END SUBROUTINE [name]
```

Programa de llamada

Subrutina

Subrutinas

```
SUBROUTINE subroutine_name ( argument_list )
```

```
...  
    (Declaration section)
```

```
...  
    (Execution section)
```

```
...  
RETURN  
END SUBROUTINE [name]
```



*Argumentos
ficticios*


```
CALL subroutine_name ( argument_list )
```



Argumentos reales

Subrutinas

```
PROGRAM piloto_de_pruebas  
  
CALL subroutine_name ( argument_list )  
  
END PROGRAM
```

```
SUBROUTINE subroutine_name ( argument_list )  
  ...  
  (Declaration section)  Variables  
  ...                     locales  
  (Execution section)  
  ...  
RETURN  
END SUBROUTINE [name]
```

Atributo INTENT

INTENT(IN)

Dummy argument is used only to pass input data to the subroutine.

INTENT(OUT)

Dummy argument is used only to return results to the calling program.

INTENT(INOUT)

or

INTENT(IN OUT)

Dummy argument is used both to pass input data to the subroutine and to return results to the calling program.

```
9      PROGRAM pitagoras
10     IMPLICIT NONE
11     REAL::cateto_ad=3., cateto_op=4., hipo
12     CALL hipotenusa(cateto_Ad, cateto_op, hipo)
13     WRITE(*,*) hipo
14     END PROGRAM pitagoras
15
16     SUBROUTINE hipotenusa(a,b,c)
17     REAL, INTENT(IN)::a,b
18     REAL, INTENT(OUT)::c
19     c=SQRT(a**2+b**2)
20     END SUBROUTINE
```

Atributo INTENT

```
16 SUBROUTINE hipotenusa(a,b,c)
17 REAL,INTENT(IN)::a,b
18 REAL,INTENT(OUT)::c
19 a=0.0
20 c=SQRT(a**2+b**2)
21 END SUBROUTINE
```

No issues found

List

Fire Solution 2 Errors 0 Warnings 0 Messages Build + IntelliSense

Code	Description	Project	File	Line
✖	Compilation Aborted (code 1)		ejemplo.f90	1
✖	error #6780: A dummy argument with the INTENT(IN) attribute shall not be defined nor become undefined. [A]		ejemplo.f90	19

Paso por Referencia



SUBROUTINE sub1

```
PROGRAM test
REAL :: a, b(4)
INTEGER :: next
...
CALL sub1 ( a, b, next )
...
END PROGRAM test
```

```
SUBROUTINE sub1 ( x, y, i )
REAL, INTENT(OUT) :: x
REAL, INTENT(IN) :: y(*)
INTEGER :: i
...
END SUBROUTINE sub1
```

(a)

Memory address	Main program name	Subroutine name
001	a	x
002	b(1)	y(1)
003	b(2)	y(2)
004	b(3)	y(3)
005	b(4)	y(4)
006	next	i
007		

Importante: Guardar consistencia con los argumentos reales y ficticios


1. Arreglo ficticio con forma explícita

```
9      PROGRAM ejemplo
10     END PROGRAM ejemplo
11
12     SUBROUTINE sub(a,b,nelement,nmax)
13     INTEGER,INTENT(IN)::nmax,nelement
14     REAL,INTENT(IN),DIMENSION(nelement)::a
15     REAL,INTENT(OUT),DIMENSION(nelement)::b
16     b(1:nmax)=2*a(1:nmax)
17     END SUBROUTINE
```

Nota: Detección de fueros de límites (Out-of-bounds) es más fácil para el compilador.

Arreglo ficticio con forma explícita

```
9  PROGRAM ejemplo
10  IMPLICIT NONE
11  REAL, DIMENSION(5)::x=(/1,2,3,4,5/),y
12  y=0.
13  CALL sub(x,y,5,3)
14  WRITE(*,*) y
15  END PROGRAM ejemplo
16
17  SUBROUTINE sub(a,b,nelement,nmax)
18  INTEGER,INTENT(IN)::nmax,nelement
19  REAL,INTENT(IN),DIMENSION(nelement)::a
20  REAL,INTENT(OUT),DIMENSION(nelement)::b
21  b(1:nmax)=2*a(1:nmax)
22  END SUBROUTINE
```




"C:\Users\cjdea\source\repos\ X + v

2.000000	4.000000	6.000000	0.000000E+00	0.000000E+00
----------	----------	----------	--------------	--------------

Press any key to continue . . .

2. Arreglo ficticio con tamaño indefinido

```
9      PROGRAM ejemplo
10     END PROGRAM ejemplo
11
12     SUBROUTINE sub(a,b,nmax)
13     INTEGER,INTENT(IN)::nmax
14     REAL,INTENT(IN),DIMENSION(*)::a
15     REAL,INTENT(OUT),DIMENSION(*)::b
16     b(1:nmax)=2*a(1:nmax)
17     END SUBROUTINE
18
19
20
21
22
23
```

100 %  No issues found


Output

Show output from: Build

Build started...

===== Build: 0 succeeded, 0 failed, 1 up-to-date, 0 skipped =====

===== Build started at 12:30 AM and took 00.094 seconds =====

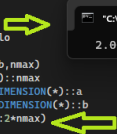


Posible Fuente de
errores lógicos

Nota: Detección de fueros de límites (Out-of-bounds) es muy difícil.

Arreglo ficticio con tamaño indefinido

```
9  PROGRAM ejemplo
10  IMPLICIT NONE
11  REAL, DIMENSION(5)::x=(/1,2,3,4,5/),y
12  y=0.
13  CALL sub(x,y,5)
14  WRITE(*,*) y
15  END PROGRAM ejemplo
16
17  SUBROUTINE sub(a,b,nmax)
18  INTEGER,INTENT(IN)::nmax
19  REAL,INTENT(IN),DIMENSION(*)::a
20  REAL,INTENT(OUT),DIMENSION(*)::b
21  b(1:3*nmax)=2*a(1:2*nmax)
22  END SUBROUTINE
```

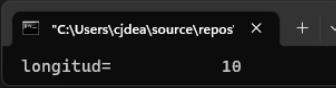


Output window: "C:\Users\cjdea\source\repos" x + v

2.000000	4.000000	6.000000	8.000000	10.00000
----------	----------	----------	----------	----------

Paso de Caracteres

```
9      PROGRAM ejemplo
10     IMPLICIT NONE
11     CHARACTER(10)::output
12     output="longitud="
13     CALL sub(output)
14     END PROGRAM ejemplo
15
16     SUBROUTINE sub(mensaje)
17     CHARACTER(*),INTENT(IN)::mensaje
18     WRITE(*,*) mensaje,Len(mensaje)
19     END SUBROUTINE
20
```



Cuando se llama la subrutina, la longitud del caracter ficticio será igual a la longitud del argumento real que se pasa desde el programa de llamada.

Programa unidad con definiciones y valores iniciales de un conjunto de datos para compartir con otros programas unidad.

```
9      MODULE geometria
10     IMPLICIT NONE
11     REAL,PARAMETER::PI=4.0*ATAN(1.0)
12     REAL, DIMENSION(5)::radios=(/1.,2.,3.,4.,5./)
13     END MODULE geometria
14
15     PROGRAM areas
16     USE geometria
17     IMPLICIT NONE
18     INTEGER::i
19     CALL calcular_areas()
20     radios=(/2.,4.,6.,8.,10./)
21     CALL calcular_areas()
22     END PROGRAM areas
23
24     SUBROUTINE calcular_areas()
25     USE geometria
26     IMPLICIT NONE
27     INTEGER::i
28     WRITE(*,*) "AREAS= ",(PI*radios(i)**2,i=1,5)
29     END SUBROUTINE
```

En el encabezado

"C:\Users\cjdea\source\repos" X + v					
AREAS=	3.141593	12.56637	28.27433	50.26548	78.53982
AREAS=	12.56637	50.26548	113.0973	201.0619	314.1593

Los módulos también pueden contener subprogramas.

```
MODULE my_subs  
  IMPLICIT NONE
```

(Declare shared data here)

```
CONTAINS
```

```
  SUBROUTINE sub1 ( a, b, c, x, error )
```

```
    IMPLICIT NONE
```

```
    REAL, DIMENSION(3), INTENT(IN) :: a
```

```
    REAL, INTENT(IN) :: b, c
```

```
    REAL, INTENT(OUT) :: x
```

```
    LOGICAL, INTENT(OUT) :: error
```

```
    ...
```

```
  END SUBROUTINE sub1
```

```
END MODULE my_subs
```

```
PROGRAM main_prog
```

```
  USE my_subs
```

```
  IMPLICIT NONE
```

```
  ...  
  CALL sub1 ( a, b, c, x, error )
```

```
  ...  
END PROGRAM main_prog
```



Interface explícita

```
9      MODULE mimodule
10     CONTAINS
11     SUBROUTINE calcular_area(a,b)
12     IMPLICIT NONE
13     REAL::a,b
14     WRITE(*,*) "AREA= ",a*b
15     END SUBROUTINE
16     END MODULE
17
18     PROGRAM area
19     USE mimodule
20     IMPLICIT NONE
21     REAL::largo
22     INTEGER::ancho
23     CALL calcular_area(largo,ancho)
24     END PROGRAM area
```

100 % No issues found

Error List

Entire Solution 2 Errors 0 Warnings 0 Messages Build + IntelliSense

	Code	Description	Project	File	Line
		Compilation Aborted (code 1)		ejemplo.f90	1
		error #6633: The type of the actual argument differs from the type of the dummy argument. [ANCHOR]		ejemplo.f90	23

- ▶ Funciones Intrínsecas
- ▶ Funciones definidas por el usuario

```
FUNCTION name ( argument_list )  
    ...  
    (Declaration section must declare type of name)  
    ...  
    (Execution section)  
    ...  
    name = expr  
    RETURN  
END FUNCTION [name]
```

1) INTEGER FUNCTION my_function (i, j)
2) FUNCTION my_function (i, j)
 INTEGER :: my_function

Definiendo el tipo

Importante: Usar INTENT(IN) para aquellos argumentos ficticios que no se modifican.

Funciones

```
9      PROGRAM area
10     IMPLICIT NONE
11     REAL::a=2.,b=3.,calcular_area,calcular_area2
12     WRITE(*,*) calcular_area(a,b)
13     WRITE(*,*) calcular_area2(a,b)
14     END PROGRAM area
15
16     FUNCTION calcular_area(largo,ancho)
17     REAL::calcular_area
18     REAL,INTENT(IN)::largo,ancho
19     calcular_area=largo*ancho
20     END FUNCTION
21
22     FUNCTION calcular_area2(largo,ancho)
23     REAL,INTENT(IN)::largo,ancho
24     calcular_area2=largo*ancho
25     END FUNCTION
```



```
"C:\Users\cidea\source\repos" X + v
6.000000
6.000000
Press any key to continue . . .
```

Subprogramas como argumentos

Los subprogramas pueden colocarse como argumentos solo si son declarados como **EXTERNAL** en el procedimiento de llamada y en el procedimiento invocado.

```
PROGRAM :: test  
REAL, EXTERNAL :: fun_1, fun_2  
REAL :: x, y, output
```



```
...  
CALL evaluate ( fun_1, x, y, output )  
CALL evaluate ( fun_2, x, y, output )  
...  
END PROGRAM test
```

```
SUBROUTINE evaluate ( fun, a, b, result )  
REAL, EXTERNAL :: fun  
REAL, INTENT(IN) :: a, b  
REAL, INTENT(OUT) :: result  
result = b * fun(a)  
END SUBROUTINE evaluate
```



Ejemplo

Escribir un programa que realice la evaluación de una función compuesta utilizando una subrutina que tome dos procedimientos externos (funciones) como argumentos.

```
PROGRAM funciones_externas
IMPLICIT NONE
EXTERNAL f,g
REAL::x,fcomp

WRITE(*,*) "Introduzca x"
READ(*,*) x
CALL funcioncompuesta(f,g,x,fcomp)
WRITE(*,*) "f(g(",x,"))=",fcomp
END PROGRAM funciones_externas
```



```
SUBROUTINE funcioncompuesta(f,g,x0,res)
IMPLICIT NONE
REAL::res,x0,f,g
res=f(g(x0))
END SUBROUTINE
```



```
REAL FUNCTION f(x)
IMPLICIT NONE
REAL,INTENT(IN)::x
f=x**2-LOG(x)
END FUNCTION
```



```
REAL FUNCTION g(x)
IMPLICIT NONE
REAL,INTENT(IN)::x
g=2*x
END FUNCTION
```

Ejemplo

Escribir un programa que realice la evaluación de una función compuesta utilizando una subrutina que tome dos procedimientos externos (subrutinas) como argumentos.

```
PROGRAM funciones_externas
IMPLICIT NONE
EXTERNAL f,g
REAL::x,fcomp

WRITE(*,*) "Introduzca x"
READ(*,*) x
CALL funcioncompuesta(f,g,x,fcomp)
WRITE(*,*) "f(g(",x,")")=",fcomp
END PROGRAM funciones_externas
```



```
SUBROUTINE funcioncompuesta(f,g,x0,res)
IMPLICIT NONE
REAL::res,x0,geval,feval
CALL g(x0,geval)
CALL f(geval,feval)
res=feval
END SUBROUTINE
```



```
SUBROUTINE f(x,y)
IMPLICIT NONE
REAL,INTENT(IN)::x
REAL,INTENT(OUT)::y
y=x**2-LOG(x)
END SUBROUTINE
```

```
SUBROUTINE g(x,y)
IMPLICIT NONE
REAL,INTENT(IN)::x
REAL,INTENT(OUT)::y
y=2*x
END SUBROUTINE
```

Escribir un programa que realice la composición de tres funciones arbitrarias $f(x)$, $g(x)$ y $h(x)$, esto es, $f(g(h(x)))$ usando procedimientos como argumentos de otros.