

# Iniciando con Fortran

Hugo de Jesús Valenzuela Chaparro

24 de febrero de 2015

## 1. Area de un circulo

Con este programa se calcula el área de un círculo de radio R que el usuario indica. Aquí un ejemplo:

```
! Area . f90 : Calcula el area de un circulo

!

Program Circle_area ! Comenzar programa

  Implicit None ! Declaracion de variables

  Real  :: radius , circum , area ! Declarar reales

  Real,PARAMETER  :: PI = 4.0 * atan(1.0) ! Declarar constante real

  Integer :: model_n = 1 ! Declare , assign Ints

  print*, "Escribe el radio del circulo:" ! hablar al usuario

  read*, radius ! leer radio

  circum = 2.0*PI*radius ! calcular circunferencia

  area = radius*radius*PI ! calcular
```

```

print*, "Numero de programa =" , model_n ! Print program number

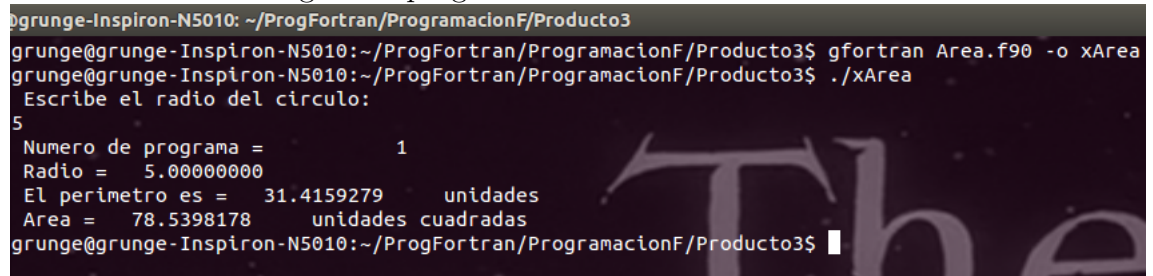
print*, "Radio =" , radius ! imprimir radio

print*, "El perimetro es =" , circum , "unidades" ! imprimir perimetro

print*, "Area =" , area , "unidades cuadradas" ! imprimir area
End Program Circle_area ! Terminar programa

```

A continuación una imagen del programa corriendo:



```

grunge-Inspiron-N5010: ~/ProgFortran/ProgramacionF/Producto3
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ gfortran Area.f90 -o xArea
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ ./xArea
Escribe el radio del circulo:
5
Numero de programa =          1
Radio =  5.00000000
El perimetro es =  31.4159279    unidades
Area =  78.5398178    unidades cuadradas
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$

```

## 2. Volumen tanque esférico

En este programa se calculó el volumen de agua en un tanque esférico de radio  $R$ , de acuerdo a la altura que este el agua respecto al fondo. Ambos datos fueron introducidos por el usuario. Aquí el código:

```

! Volumen.f90 : Calcula el volumen del agua que se encuentra
! en un tanque esferico a una altura h del suelo
!

```

```

Program Volumen_esfera ! comenzar programa

```

```

Implicit None ! Declaracion de variables

```

```

Real :: radio , volumen , altura ! Declarar reales

```

```

Real,PARAMETER :: PI = 4.0 * atan(1.0) ! Declarar constaste

```

```

Integer :: model_n = 2 ! Declarar entero con valor dado

```

```

!Pedir datos
! hablar al usuario
print*, "Escribe el radio de la esfera (mayor o igual que cero):" o
read*, radio
print*, "Escribe la altura a la que esta el agua desde el piso &
& o fondo del tanque esferico"
read*, altura

```

```

IF (altura <= (2.0*radio)) THEN
    ! calcular el volumen del agua en el tanque
    volumen = (PI/3.0)*(altura*altura)*((3.0*radio)-altura)

```

```

ELSE
    print*, "Verifica que los datos sean consistentes"
    STOP
END IF

```

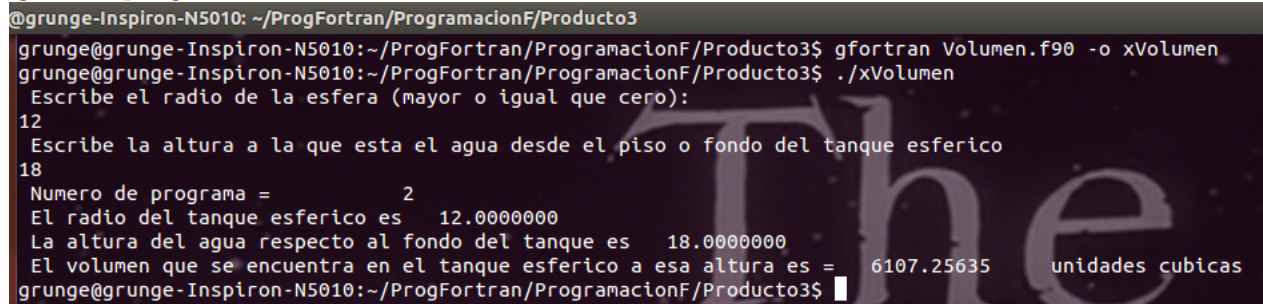
```

print*, "Numero de programa =" , model_n ! Print program number
print*, "El radio del tanque esferico es" , radio !radio
print*, "La altura del agua respecto al fondo del tanque es" , altura !altura
print*, "El volumen que se encuentra en el tanque esferico a &
& esa altura es =" , volumen , "unidades cubicas"

```

End Program Volumen\_esfera ! Terminar programa

Imagen del programa corriendo:



```

@grunge-Inspiron-N5010: ~/ProgFortran/ProgramacionF/Producto3
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ gfortran Volumen.f90 -o xVolumen
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ ./xVolumen
Escribe el radio de la esfera (mayor o igual que cero):
12
Escribe la altura a la que esta el agua desde el piso o fondo del tanque esferico
18
Numero de programa =      2
El radio del tanque esferico es  12.0000000
La altura del agua respecto al fondo del tanque es  18.0000000
El volumen que se encuentra en el tanque esferico a esa altura es =  6107.25635 unidades cubicas
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$

```

### 3. Presición Sencilla \*4

Este programa sirve para ver la presición sencilla \*4 de la máquina en los cálculos. Aquí el código:

```
! Limits . f90 : Determina la presicion de la maquina,
! presicion sencilla *4

!

Program Limits

Implicit None

Integer :: i , n

Real *4 :: epsilon_m , one
Integer :: model_n = 41 ! Declare , assign Ints
n=60 ! Establece el numero de iteraciones

! Dar valores iniciales :

epsilon_m = 1.0

one = 1.0

! calcular cada paso con DO-LOOP e imprimir

! Se ejecutara 60 veces de acuerdo a i

! Incrementado de 1 a n (debido a n=60)
print*, "Numero de programa =" , model_n !Print program number
do i = 1, n , 1 ! Comienza el loop

    epsilon_m = epsilon_m / 2.0 ! reduce epsilon m

    one = 1.0 + epsilon_m ! calcula de nuevo one
```

```

    print*, i , one , epsilon_m ! imprimir valores

end do ! terminar loop cuando i>n

```

End Program Limits

Imagen del programa corriendo:

```

grunge-Inspiron-N5010: ~/ProgFortran/ProgramacionF/Producto3$ gfortran PresicionSencilla_4.f90 -o xPresicionSencilla_4
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ ./xPresicionSencilla_4
Numero de programa = 41
 1  1.50000000  0.50000000
 2  1.25000000  0.25000000
 3  1.12500000  0.12500000
 4  1.06250000  6.25000000E-02
 5  1.03125000  3.12500000E-02
 6  1.01562500  1.56250000E-02
 7  1.00781250  7.81250000E-03
 8  1.00390625  3.90625000E-03
 9  1.00195312  1.95312500E-03
10  1.00097656  9.76562500E-04
11  1.00048828  4.88281250E-04
12  1.00024414  2.44140625E-04
13  1.00012207  1.22070312E-04
14  1.00006104  6.10351562E-05
15  1.00003052  3.05175781E-05
16  1.00001526  1.52587891E-05
17  1.00000763  7.62939453E-06
18  1.00000381  3.81469727E-06
19  1.00000191  1.90734863E-06
20  1.00000095  9.53674316E-07
21  1.00000048  4.76837158E-07
22  1.00000024  2.38418579E-07
23  1.00000012  1.19209290E-07
24  1.00000000  5.96046448E-08
25  1.00000000  2.98023224E-08
26  1.00000000  1.49011612E-08
27  1.00000000  7.45058060E-09
28  1.00000000  3.72529030E-09
29  1.00000000  1.86264515E-09
30  1.00000000  9.31322575E-10
31  1.00000000  4.65661287E-10
32  1.00000000  2.32830644E-10
33  1.00000000  1.16415322E-10
34  1.00000000  5.82076609E-11
35  1.00000000  2.91038305E-11
36  1.00000000  1.45519152E-11
37  1.00000000  7.27595761E-12
38  1.00000000  3.63797881E-12

```

## 4. Presición sencilla real

En este programa se ve la presición sencilla real de la máquina. Aquí el código:

```

! Limits . f90 : Determina la presicion de la maquina,
! presicion sencilla *4

```

!

Program Limits

Implicit None

Integer :: i , n

Real :: epsilon\_m , one

Integer :: model\_n = 42 ! Declare , assign Ints  
n=60 ! Establece el numero de iteraciones

! Dar valores iniciales :

epsilon\_m = 1.0

one = 1.0

! calcular cada paso con DO-LOOP e imprimir

! Se ejecutara 60 veces de acuerdo a i

! Incrementado de 1 a n (debido a n=60)

print\*, "Numero de programa =" , model\_n ! Print program number  
do i = 1, n , 1 ! Comienza el loop

epsilon\_m = epsilon\_m / 2.0 ! reduce epsilon m

one = 1.0 + epsilon\_m ! calcula de nuevo one

print\*, i , one , epsilon\_m ! imprimir valores

end do ! terminar loop cuando i>n

End Program Limits

Imagen del programa corriendo:

```

grunge-Inspiron-N5010: ~/ProgFortran/ProgramacionF/Producto3
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ gfortran PresicionSencilla_Real.f90 -o xPresicionSencilla_Real
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ ./xPresicionSencilla_Real
Numero de programa = 42
 1  1.50000000  0.50000000
 2  1.25000000  0.25000000
 3  1.12500000  0.12500000
 4  1.06250000  6.25000000E-02
 5  1.03125000  3.12500000E-02
 6  1.01562500  1.56250000E-02
 7  1.00781250  7.81250000E-03
 8  1.00390625  3.90625000E-03
 9  1.00195312  1.95312500E-03
10  1.00097656  9.76562500E-04
11  1.00048828  4.88281250E-04
12  1.00024414  2.44140625E-04
13  1.00012207  1.22070312E-04
14  1.00006104  6.10351562E-05
15  1.00003052  3.05175781E-05
16  1.00001526  1.52587891E-05
17  1.00000763  7.62939453E-06
18  1.00000381  3.81469727E-06
19  1.00000191  1.90734863E-06
20  1.00000095  9.53674316E-07
21  1.00000048  4.76837158E-07
22  1.00000024  2.38418579E-07
23  1.00000012  1.19209290E-07
24  1.00000000  5.96046448E-08
25  1.00000000  2.98023224E-08
26  1.00000000  1.49011612E-08
27  1.00000000  7.45058060E-09
28  1.00000000  3.72529030E-09
29  1.00000000  1.86264515E-09
30  1.00000000  9.31322575E-10
31  1.00000000  4.65661287E-10
32  1.00000000  2.32830644E-10
33  1.00000000  1.16415322E-10
34  1.00000000  5.82076609E-11
35  1.00000000  2.91038305E-11
36  1.00000000  1.45519152E-11
37  1.00000000  7.27595761E-12
38  1.00000000  3.63797881E-12

```

## 5. Funciones intrínsecas

Este programa sirve para ejemplificar cómo y para qué se usan las funciones intrínsecas de Fortran. Aquí el código:

```

! Math . f90 : ejemplos de algunas funciones de Fortran

!

Program math ! Comenzar programa
  ! Declaracion de variables4
  Real *8 :: x =-1.0 , y=0.2 , z=0 , arccos, log, i, a
  Integer :: model_n = 6 ! Declare , assign Ints

a=ABS(x)

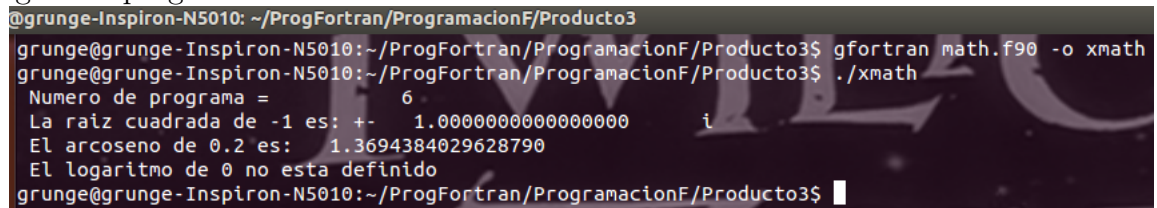
```

```
i =SQRT(a)
arccos=ACOS(y)
```

```
print*, "Numero de programa =" , model_n !Print program number
print*, "La raiz cuadrada de -1 es: +-" , i , "i"
print*, "El arcoseno de 0.2 es:" , arccos
print*, "El logaritmo de 0 no esta definido"
```

End Program math ! Terminar programa

Imagen del programa corriendo:



```
@grunge-Inspiron-N5010: ~/ProgFortran/ProgramacionF/Producto3
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ gfortran math.f90 -o xmath
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ ./xmath
Numero de programa =          6
La raiz cuadrada de -1 es: +-  1.0000000000000000      i
El arcoseno de 0.2 es:   1.3694384029628790
El logaritmo de 0 no esta definido
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$
```

## 6. Funciones definidas por el usuario

Este programa sirve para ilustrar cómo se definen funciones en Fortran y cómo se llaman en el programa principal. Aquí el código:

```
! Function . f90 : Llama a una funcion definida por el usuario
```

```
!
```

```
Real *8 Function f (x,y)
```

```
Implicit None
```

```
Real *8 :: x, y
```

```
f = 1.0 + sin (x*y )
```

```
End Function f
```



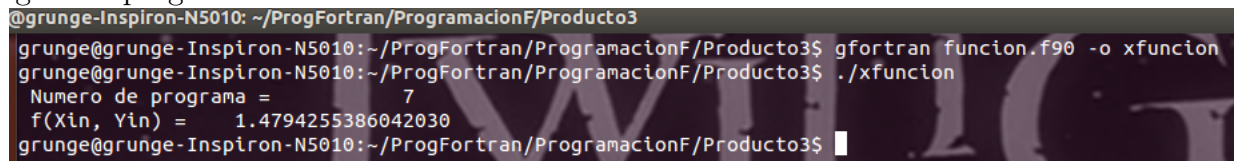
Program Main

Implicit None

```
Real *8 :: Xin =0.25 , Yin =2. , c , f ! declarations ( also f)
Integer :: model_n = 7 ! Declare , assign Ints
c = f ( Xin , Yin )
print*, "Numero de programa =" , model_n ! Print program number
write ( * , * ) "f(Xin, Yin) = " , c
```

End Program Main

Imagen del programa corriendo:



```
@grunge-Inspiron-N5010: ~/ProgFortran/ProgramacionF/Producto3
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ gfortran funcion.f90 -o xfuncion
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ ./xfuncion
Numero de programa = 7
f(Xin, Yin) = 1.4794255386042030
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$
```

## 7. Subrutinas

Este programa ilustrar cómo se hace una subrutina y se llama en el programa principal. Aquí en código:

```
! Subroutine . f90 : Muestra como se llama una subrutina
!
```

```
Subroutine g(x, y, ans1 , ans2 )
```

Implicit None

```
Real (8) :: x , y , ans1 , ans2 ! Declarar variables
```

```
ans1 = sin (x*y) + 1. ! Usar funcion intrinseca
```

```
ans2 = ans1**2
```

```
End Subroutine g
```

```
!
```

```
Program Main ! Demos the CALL
```

```
Implicit None
```

```
Real *8 :: Xin =0.25 , Yin =2.0 , Gout1 , Gout2
```

```
Integer :: model_n = 8 ! Declare , assign Ints
```

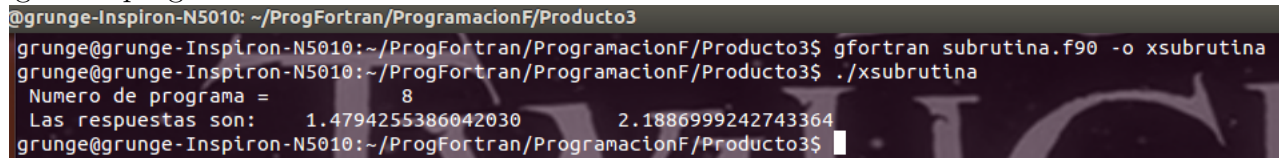
```
call g( Xin , Yin , Gout1 , Gout2 ) ! Llamar la subrutina
```

```
print*, "Numero de programa =" , model_n !Print program number
```

```
write ( * , *) "Las respuestas son: " , Gout1 , Gout2
```

```
End Program Main
```

Imagen del programa corriendo:



```
@grunge-Inspiron-N5010: ~/ProgFortran/ProgramacionF/Producto3
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ gfortran subrutina.f90 -o xsubrutina
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$ ./xsubrutina
Numero de programa =      8
Las respuestas son:    1.4794255386042030      2.1886999242743364
grunge@grunge-Inspiron-N5010:~/ProgFortran/ProgramacionF/Producto3$
```