

# Iniciando con Fortran.

Olga María Fimbres Morales.

1.- Programa Area.f90, para calcular el área de un círculo (Sección 9.3, pág. 201).

```
! Area . f90 : Calculates the area of a circle, sample program
!  
Program Circle_area ! Begin main program  
  Implicit None ! Declare all variables  
  Real 8 :: radius , circum , area ! Declare Reals  
  Real 8 :: PI = 4.0 * atan(1.0) ! Declare , assign Real  
  Integer :: model_n = 1 ! Declare , assign Ints  
  print  , 'Enter a radius:' ! Talk to user  
  read  , radius ! Read into radius  
  circum = 2.0 * PI * radius ! Calc circumference  
  area = radius * radius * PI ! Calc area  
  print  , 'Program number =' , model_n ! Print program number  
  print  , 'Radius =' , radius ! Print radius  
  print  , 'Circumference =' , circum ! Print circumference  
  print  , 'Area =' , area ! Print area  
End Program Circle_area ! End main program code
```

Captura este programa con el editor Emacs, compila y ejecuta el binario:  
gfortran Area.f90 -o xArea ./xArea

2.- Modifica el programa Area.f90 y crea un nuevo programa fuente Volumen.f90 que calcula el volumen de líquido en un tanque esférico de radio  $R$ , en el caso en que el nivel de líquido se encuentre a una altura  $H$  medida desde el fondo del tanque. Ver esta referencia en Wolfram para deducir la fórmula.

Para la realización de este ejercicio, fue necesario cambiar los datos que el programa utiliza para realizar la orden, es decir, cambiar el área y la circunferencia por la altura y el volumen. Tras hacer esto, se debe especificar al programa que pida al usuario estos datos, para que pueda ser capaz de realizar el cálculo con la fórmula correspondiente. Para finalmente proporcionar el resultado.

```

! Area . f90 : Calculates the area of a circle, sample program
!
Program Circle_volume ! Begin main program
  Implicit None ! Declare all variables
  Real *8 :: radio , altura ,volumen ! Declare Reals
  Complex *8 :: PI = 4.0 * atan(1.0) ! Declare , assign Real
  Integer :: model_n = 1 ! Declare , assign Ints
  print * , 'Ingrese un radio:' ! Talk to user
  read * , radio
  print * , 'Ingrese una altura:'
  read * , altura
  volumen = (2.0/3.0) * PI * radio * radio * altura ! Calc are
  print * , 'Volumen =' , volumen
End Program Circle_volume

```

```

terminal
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$ gfortran Voluman.f90 -o
volx
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$ ./volx
  Ingrese un radio:
4
  Ingrese una altura:
5
  Volumen = 167.55161285400391
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$ █

```

3.- Determinando la precisión de la máquina. Se proporciona el siguiente código de doble precisión (Sección 10.7, pág. 212):

```
! Limits . f90 : Determines machine precision
!
Program Limits
  Implicit None
  Integer :: i , n
  Real 8 :: epsilon_m , one
  n=60 ! Establish the number of iterations
  ! Set initial values :
  epsilon_m = 1.0
  one = 1.0
  ! Within a DOLLOOP, calculate each step and print .
  ! This loop will execute 60 times in a row as i is
  ! incremented from 1 to n ( since n = 60) :
  do i = 1, n , 1 ! Begin the doloop
    epsilon_m = epsilon_m / 2.0 ! Reduce epsilon m
    one = 1.0 + epsilon_m ! Recalculate one
    print , i , one , epsilon_m ! Print values so far
  end do ! End loop when i>n
End Program Limits
```

4.- Modifica el programa anterior para realizar las operaciones en precisión sencilla: real \*4 o simplemente real.

Al modificar el valor de Real”hacemos que la computadora nos muestre una menor aproximación que si la hicieramos con Real\*8”.

```
! Limits . f90 : Determines machine precision
!
Program Limits
  Implicit None
  Integer :: i , n
  Real :: epsilon_m , one
  n=60 ! Establish the number of iterations
  ! Set initial values :
  epsilon_m = 1.0
  one = 1.0
! Within a DOLLOOP, calculate each step and print .
! This loop will execute 60 times in a row as i is
! incremented from 1 to n ( since n = 60) :
  do i = 1, n , 1 ! Begin the doloop
```

```

epsilon_m = epsilon_m / 2.0 ! Reduce epsilon m
one = 1.0 + epsilon_m ! Recalculate one
print * , i , one , epsilon_m ! Print values so far
end do ! End loop when i>n
End Program Limits

```

```

omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$ gfortran pres.f90 -o pre
SX
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$ ./presx

```

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
39	1.00000000	1.81898940E-12
40	1.00000000	9.09494702E-13
41	1.00000000	4.54747351E-13
42	1.00000000	2.27373675E-13
43	1.00000000	1.13686838E-13
44	1.00000000	5.68434189E-14
45	1.00000000	2.84217094E-14
46	1.00000000	1.42108547E-14

5.- Fortran maneja las funciones trigonométricas y las especiales, mas no maneja relaciones entre ellas (Sección 10.9, pág. 215).

```
! Math . f90 : demo some Fortran math functions
!  
Program Math test ! Begin main program  
  Real 8 :: x = 1.0 , y, z ! Declare variables x, y, z  
  y = sin (x) ! Call the sine function  
  z = exp (x) + 1.0 ! Call the exponential function  
  print  , x, y, z ! Print x, y, z  
End Program Math test ! End main program
```

6.- Modifica el programa anterior Math.f90, para calcular, la raíz cuadrada de -1; el arcoseno de 2.0; y el logaritmo de 0.

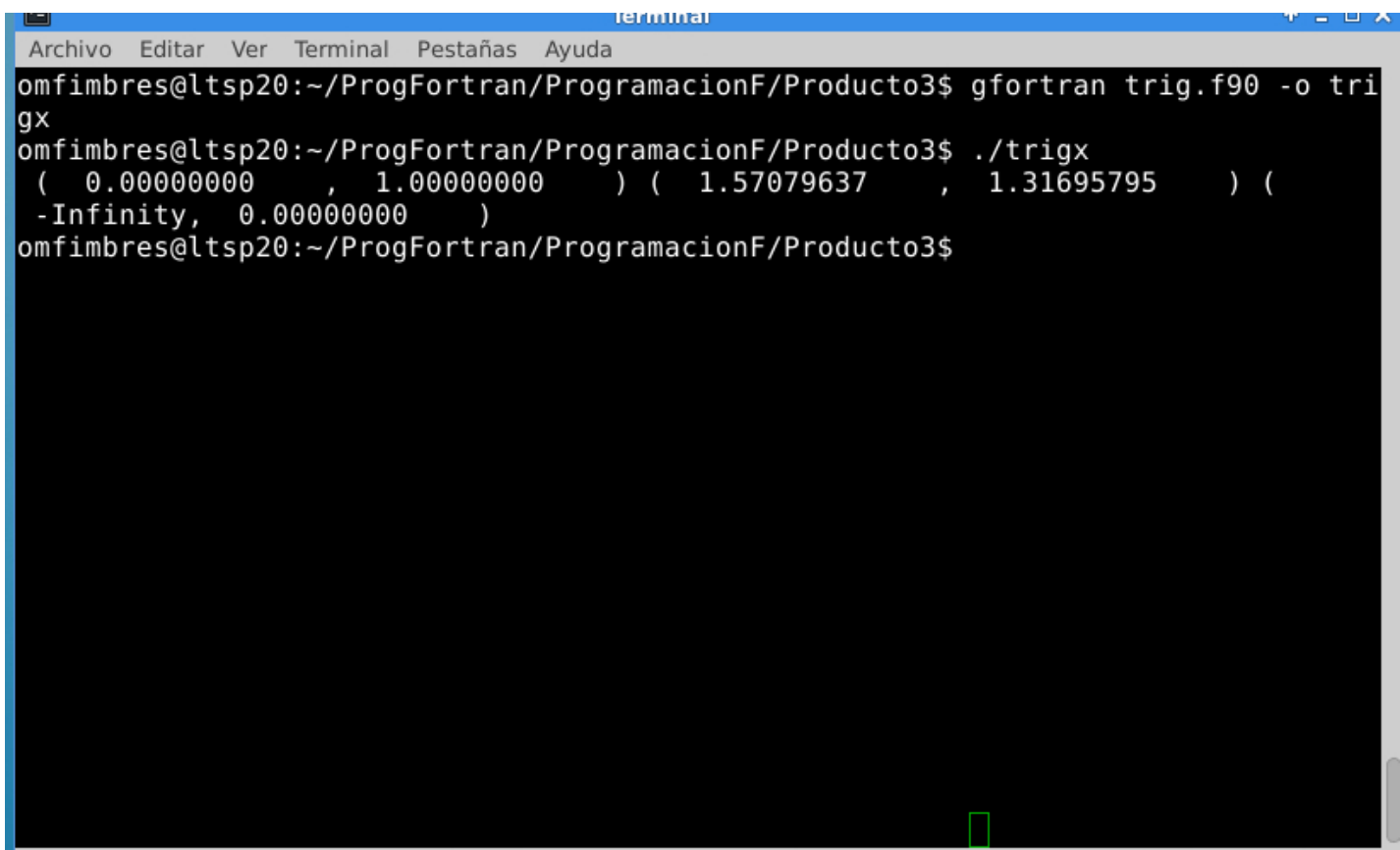
Primeramente debemos cambiar los valores reales por complejos, ya que los resultados que estamos buscando son de esta naturaleza. Seguidamente, debemos especificar aquellas funciones que estamos buscando, es decir la raíz cuadrada la cual se especifica con "sqrt"; el arcoseno, al que le corresponde ".asin"; y finalmente la funcion logaritmo, la que especificamos con "log".

Una vez hecho esto, basta con especificar los valores a los que deseamos aplicar estas funciones.

```

! Math . f90 : demo some Fortran math functions
!
Program Math_test ! Begin main program
  Complex *8 :: x=-1.0 , y=2.0, z=0.0 ! Declare variables x, y, z
  x = sqrt (x)
  y = asin (y) ! Call the sine function
  z = log (z) ! Call the exponential function
  print * , x, y, z ! Print x, y, z
End Program Math_test ! End main program

```



The screenshot shows a terminal window with a blue title bar labeled "terminal". The menu bar includes "Archivo", "Editar", "Ver", "Terminal", "Pestañas", and "Ayuda". The terminal content shows the following commands and output:

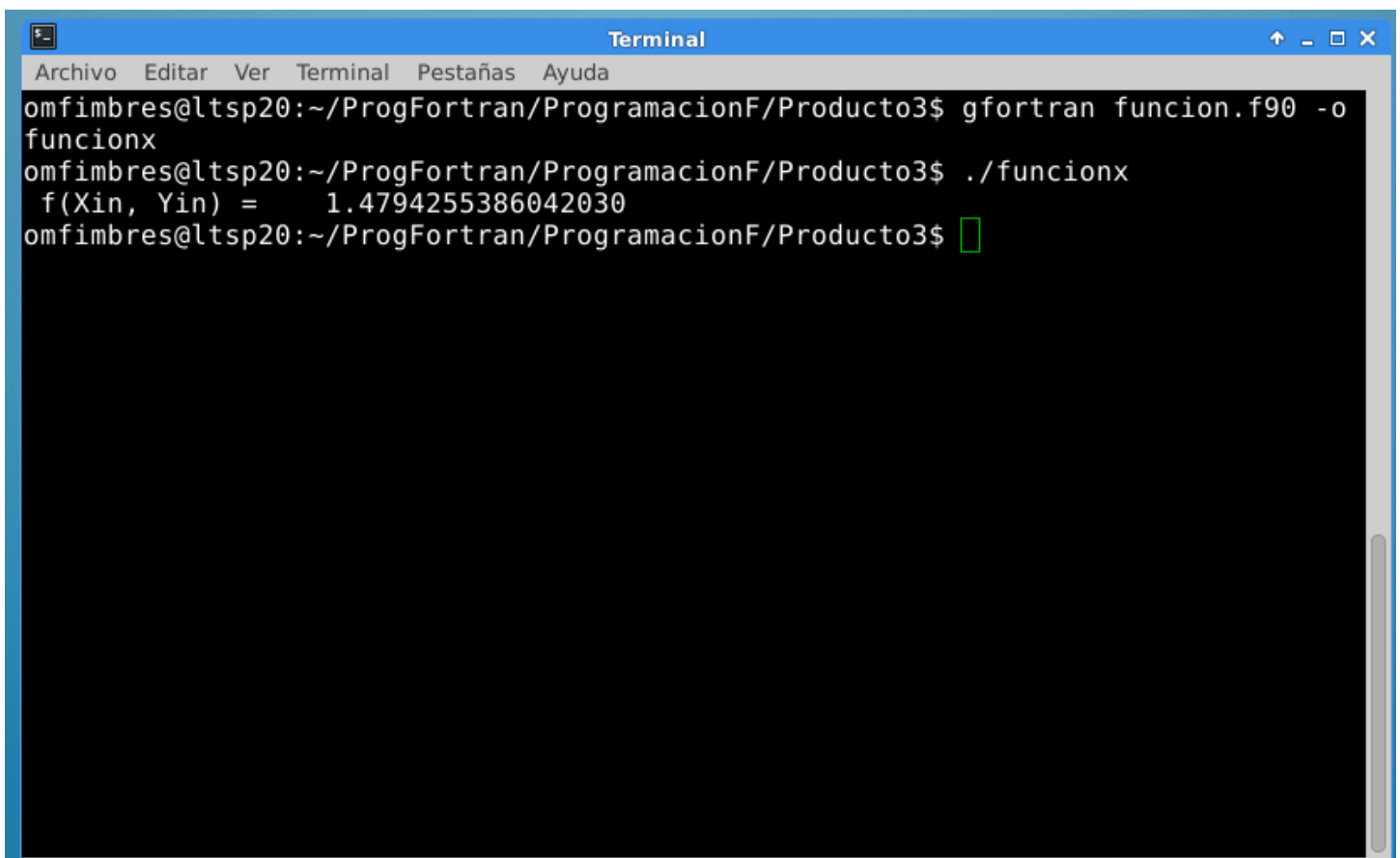
```

omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$ gfortran trig.f90 -o trigx
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$ ./trigx
( 0.00000000 , 1.00000000 ) ( 1.57079637 , 1.31695795 ) (
-Infinity, 0.00000000 )
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$

```

7.- Compila y ejecuta el siguiente programa calcula el valor de una función  $f(x, y) = 1 + \sin(x y)$  definida por el usuario (Sección 10.10, pág. 217).

```
! Function . f90 : Program calls a simple function
!
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)
  c= f(Xin , Yin)
  write ( * , * ) 'f(Xin, Yin) = ' , c
End Program Main
```

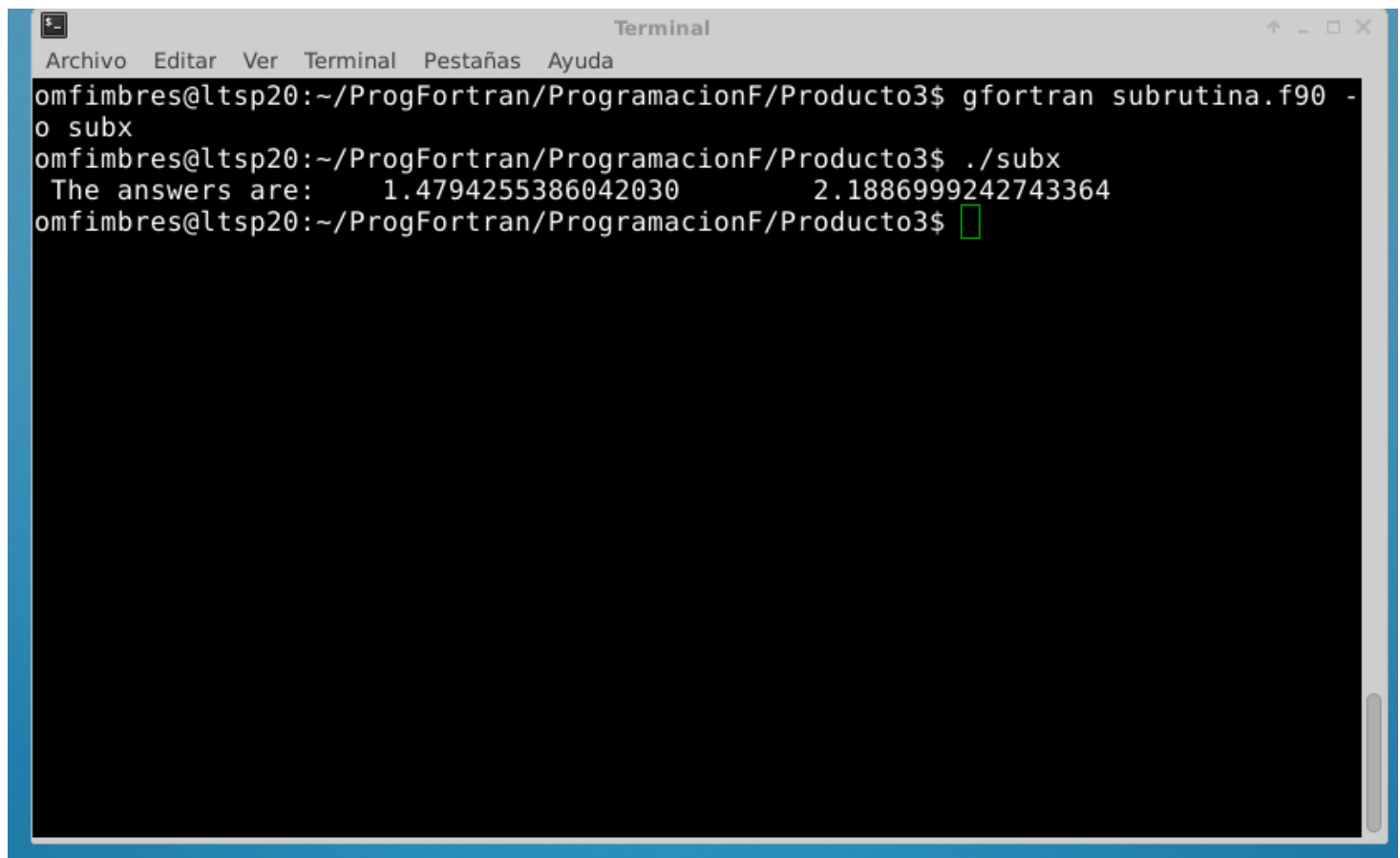


The screenshot shows a terminal window titled "Terminal" with a menu bar containing "Archivo", "Editar", "Ver", "Terminal", "Pestañas", and "Ayuda". The terminal content shows the following commands and output:

```
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$ gfortran funcion.f90 -o
funcionx
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$ ./funcionx
f(Xin, Yin) =      1.4794255386042030
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$
```

8.- Fortran además de funciones, también se manejan subrutinas. El siguiente programa contiene un ejemplo de una subrutina (Sección 10.10, pág. 218)

```
! Subroutine . f90 : Demonstrates the call for a simple subroutine
!
Subroutine g(x, y, ans1 , ans2 )
  Implicit None
  Real (8) :: x , y , ans1 , ans2 ! Declare variables
  ans1 = sin (x*y) + 1. ! Use sine i n t r i n s i c func .
  ans2 = ans1 **2
End Subroutine g
!
Program Main_program ! Demos the CALL
  Implicit None
  Real *8 :: Xin =0.25 , Yin =2.0 , Gout1 , Gout2
  call g( Xin , Yin , Gout1 , Gout2 ) ! Call the subr g
  write ( * , *) 'The answers are: ' , Gout1 , Gout2
End Program Main_program
```



```
Terminal
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$ gfortran subrutina.f90 -o subx
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$ ./subx
The answers are: 1.4794255386042030 2.1886999242743364
omfimbres@ltsp20:~/ProgFortran/ProgramacionF/Producto3$
```