

Reporte de Actividad 3

Carlos Medina

20-2-15

El siguiente reporte describirá los pasos realizados para la actividad 3 (2015-1), así como ilustrará los resultados de ésta.

Fortran 90 (previamente FORTRAN) es un lenguaje de programación alto nivel de propósito general, procedimental e imperativo, que está especialmente adaptado al cálculo numérico y a la computación científica. Es el sucesor de FORTRAN 77, informalmente conocido como Fortran 90 (y anterior a eso, Fortran 8X), fue finalmente lanzado como ISO/IEC estándar en 1991 y un ANSI estándar en 1992. En adición de cambiar el nombre oficial de FORTRAN a Fortran, además de agregar muchas características nuevas para reflejar los cambios significativos en la práctica de la programación que ha evolucionado desde el estándar de 1978.

A continuación veremos unos cálculos que se resolverán utilizando Fortran 90, de calcular áreas a funciones. Primero empezaremos calculando

el área de un círculo en función del radio que tú elijas:

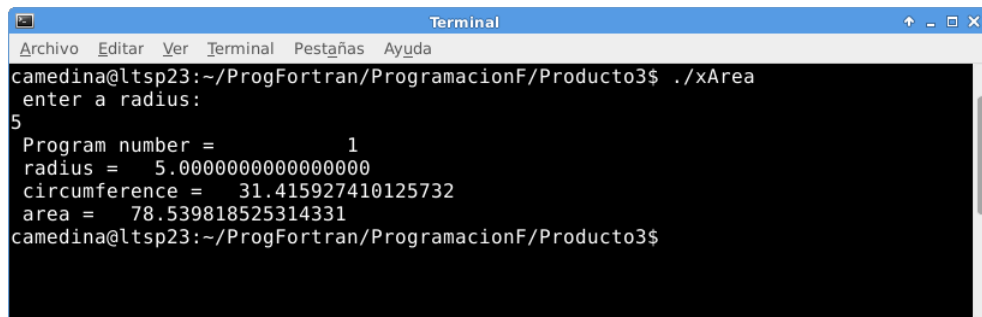
```
Program circle_area
Implicit None
Real *8 :: radius, circum, area
Real *8 :: PI = 4.0 * atan(1.0)
Integer :: model_n = 1
print *, 'enter a radius:'
read (*,*) radius
circum = 2.0 * PI * radius
area = radius * radius * PI
print *, 'Program number =' , model_n
```

```

print *, 'radius =' , radius
print *, 'circumference =' , circum
print *, 'area =' , area
end Program circle_area

```

Después de compilar, procedemos a ejecutar el programa y escribir el radio que deseamos a nuestro gusto, entonces nos saldrá la siguiente pantalla:



```

Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$ ./xArea
enter a radius:
5
Program number =          1
radius =    5.0000000000000000
circumference =    31.415927410125732
area =    78.539818525314331
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$

```

Ahora probaremos con otro cálculo, por ejemplo, calcular 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. Para esto, establecemos las variables y las formulas, después, tendremos este código:

```

Program circle_area
Implicit None
Real *8 :: radius, high, difference, volume
Real *8 :: PI = 4.0 * atan(1.0)
Integer :: model_n = 1

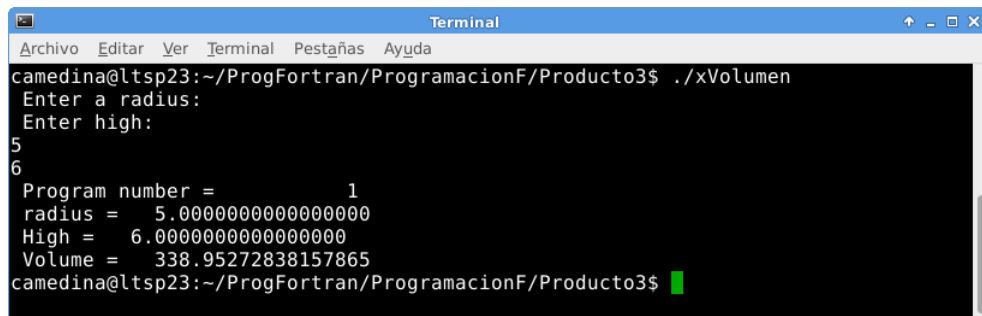
```

```

print *, 'Enter a radius:'
print *, 'Enter high:'
read (*,*) radius
read (*,*) high
diference = 3 * radius - high
volume = 0.333 * PI * high * high * diference
print *, 'Program number =' , model_n
print *, 'radius =' , radius
print *, 'High =' , high
print *, 'Volume =' , volume
end Program circle_area

```

Se puede observar el uso de las variables reales como lo son el radio, la altura, para que te calcule el volumen. cabe recalcar que la variable "diference" fue agregada para facilitar el cálculo acorde con la fórmula. Procedemos a compilarlo y cuando lo ejecutemos, saldrá el siguiente resultado:



```

Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$ ./xVolumen
Enter a radius:
Enter high:
5
6
Program number =          1
radius =  5.0000000000000000
High =   6.0000000000000000
Volume =  338.95272838157865
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$

```

Ahora, para el siguiente paso, se proporciona el siguiente código de doble precisión. Este código podrá a prueba los decimales que queramos para mejorar o simplificar la precisión.

```

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

```

Este código mostrará un valor que consta de muchísimas decimales, y con la precisión asignada, *8, te muestra un total de 16 decimales, como se ve en la siguiente imagen.

```

Terminal
Archivo Editar Ver Terminal Pestañas Ayuda

camedina@lts23:~/ProgFortran/ProgramacionF/Producto3$ ./xLimites
 1  1.5000000000000000  0.5000000000000000
 2  1.2500000000000000  0.2500000000000000
 3  1.1250000000000000  0.1250000000000000
 4  1.0625000000000000  6.250000000000000E-002
 5  1.0312500000000000  3.125000000000000E-002
 6  1.0156250000000000  1.562500000000000E-002
 7  1.0078125000000000  7.812500000000000E-003
 8  1.0039062500000000  3.906250000000000E-003
 9  1.0019531250000000  1.953125000000000E-003
10  1.0009765625000000  9.765625000000000E-004
11  1.0004882812500000  4.882812500000000E-004
12  1.0002441406250000  2.441406250000000E-004
13  1.0001220703125000  1.220703125000000E-004
14  1.0000610351562500  6.103515625000000E-005
15  1.0000305175781250  3.051757812500000E-005
16  1.0000152587890625  1.525878906250000E-005
17  1.0000076293945312  7.629394531250000E-006
18  1.0000038146972656  3.814697265625000E-006
19  1.0000019073486328  1.907348632812500E-006
20  1.0000009536743164  9.536743164062500E-007
21  1.0000004768371582  4.768371582031250E-007
22  1.0000002384185791  2.384185791015625E-007
23  1.0000001192092896  1.192092895507812E-007
24  1.0000000596046448  5.960464477539062E-008
25  1.0000000298023224  2.980232238769531E-008
26  1.0000000149011612  1.490116119384765E-008
27  1.0000000074505806  7.450580596923828E-009
28  1.0000000037252903  3.725290298461914E-009
29  1.0000000018626451  1.862645149230957E-009
30  1.0000000009313226  9.313225746154785E-010
31  1.0000000004656613  4.656612873077392E-010
32  1.0000000002328306  2.328306436538696E-010
33  1.0000000001164153  1.164153218269348E-010
34  1.0000000000582077  5.820766091346740E-011
35  1.0000000000291038  2.910383045673370E-011
36  1.0000000000145519  1.455191522836685E-011
37  1.0000000000072760  7.275957614183425E-012
38  1.0000000000036380  3.637978807091713E-012
39  1.0000000000018190  1.818989403545856E-012
40  1.0000000000009095  9.094947017729282E-013
41  1.0000000000004547  4.547473508864641E-013
42  1.0000000000002274  2.273736754432320E-013
43  1.0000000000001137  1.136868377216160E-013
44  1.0000000000000568  5.684341886080801E-014
45  1.0000000000000284  2.842170943040400E-014
46  1.0000000000000142  1.421085471520200E-014

```

Ahora, tratemos con cantidad menor de decimales, cambiando el valor *Real*8* por *Real*4* o simplemente *Real* y así, nos dará 8 decimales solamente.

```

! Limits . f90 : Determines machine precision
! -----
Program Limits
  Implicit None

```

```

Integer :: i , n
Real *4 :: epsilon_m , one
n=60 ! Establish the number of iterations
! Set initial values :
epsilon_m = 1.0
one = 1.0
! Within a DO-LOOP, 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 do-loop
    epsilon_m = epsilon_m / 2.0 ! Reduce epsilon m
    one = 1.0 + epsilon_m ! Re-calculate one
    print * , i , one , epsilon_m ! Print values so far
end do ! End loop when i>n
End Program Limits

```

Compilamos, ejecutamos, y nos aparecerá:

```
Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$ ./xLmites2
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
47 1.00000000 7.10542736E-15
```

Como podemos ver, cuando determinamos la precisión, nos da el doble del número que elegimos (*8 para 16, *4 para 8).

Fortran también maneja las funciones trigonométricas y las especiales. A continuación veremos un ejemplo de la función seno y la función exponencial.

```
!Math . f90 : demo some Fortran math functions
```

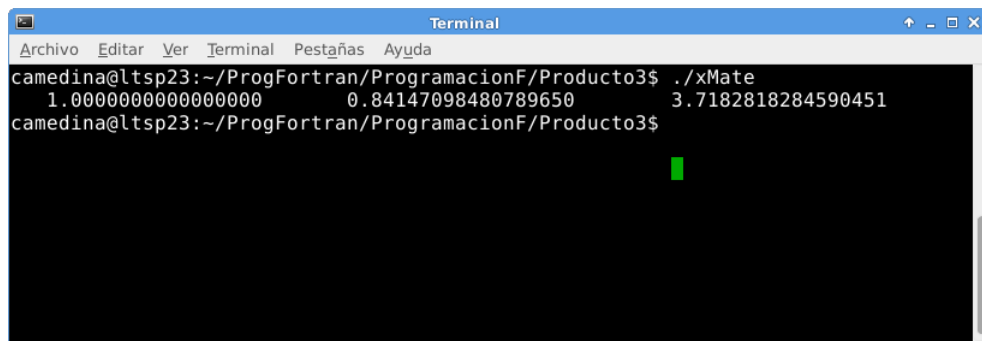
```
! -----
```

```

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

```

Compilamos y nos saldrá el resultado.



```

Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$ ./xMate
1.0000000000000000 0.84147098480789650 3.7182818284590451
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$

```

Modificaremos el programa anterior para calcular la raíz cuadrada de -1; el arcoseno de 2.0; y el logaritmo de 0.

```

!Math . f90 : demo some Fortran math functions
! -----
Program Math_test ! Begin main program
  Real *8 :: x = -1.0 , y=2.0, z=0 ! Declare variables x, y, z, a, b, c
  a = sqrt (x) ! Call the sine function
  b = asin (y) ! Call the exponential function
  c = log (z)
  print * , x, y, z, a, b, c ! Print x, y, z
End Program Math_test ! End main program

```

Como se puede observar, se declaran las variables x, y, z (valores) y a, b, c (funciones), escribiendo después la orden para imprimir los resultados. Compilando y ejecutando se muestra lo dicho.


```

Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$ ./xMate2
-1.0000000000000000      2.0000000000000000      0.0000000000000000
      NaN      NaN      -Infinity
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$

```

También es posible calcular el valor de una función $f(x, y) = 1 + \sin(x \cdot y)$ definida por el usuario, escribiendo el siguiente código, compilándolo y ejecutándolo:

```

! 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

```

```

Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$ ./xFuncion
f(Xin, Yin) = 1.4794255386042030
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$

```

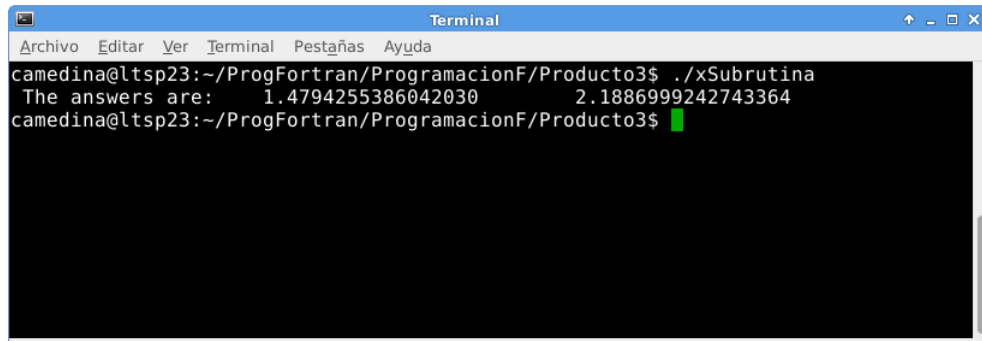
Fortran puede hacer bastantes calculos, y además de funciones, también se manejan subrutinas. Probaremos escribiendo el siguiente código:

```

! 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 intrinsic 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

```

Por último, compilamos y ejecutamos estos comandos.



```

Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$ ./xSubrutina
The answers are:      1.4794255386042030      2.1886999242743364
camedina@ltsp23:~/ProgFortran/ProgramacionF/Producto3$

```