

Iniciando Fortran

Martín Alejandro Paredes Sosa

Febrero 2015

1. Introducción

En esta práctica se inicio a trabajar con **FORTRAN**, escribiendo algunos comandos y programas sencillos. Estos nos permiten entender la sintaxis y así poder saber como utilizar los comandos de Fortran.

En el siguiente documento se mostraran los trabajos que se realizaron durante esta actividad así como los resultados de ejecutar estos programas.

2. Actividades Realizadas

A continuación se mostraran los programas que se realizaron durante esta actividad. Se mostraran con el siguiente formato:

- Descripción del programa
- El código en Fortran
- Una imagen de la ejecución del programa

2.1. Programa: Área del Circulo

En este programa se pide al usuario que proporcione el radio del circulo con el cual se realizaran los cálculos de la circunferencia y área del circulo.

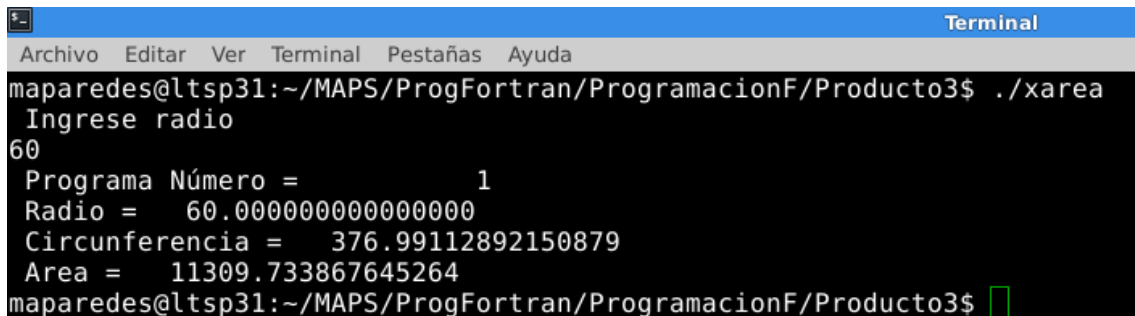
```
!=====
! Area.f90 : Calcula el area del circulo
!=====

program Area_Circulo !Inicio de programa

  Implicit None
  Real *8 :: radius , circum , area !Declaracion de variables
  Real *8 :: PI = 4.0*atan(1.0)
  Integer :: model_n =1
  print *, 'Ingrese radio' !Hablar con el usuario
  read *, radius !Leer respuesta de usuario
  circum = 2.0*PI*radius !Calculo de la circunferencia
  area = radius*radius*PI !Calcula el area
  print *, 'Programa Número =', model_n
  print *, 'Radio =', radius
  print *, 'Circunferencia =' , circum
  print *, 'Area =', area

end program Area_Circulo
```

Ejecución del Programa



```
Terminal
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$ ./xarea
Ingrese radio
60
Programa Número =          1
Radio =    60.000000000000000
Circunferencia =   376.99112892150879
Area =   11309.733867645264
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$ █
```

2.2. Programa: Volumen

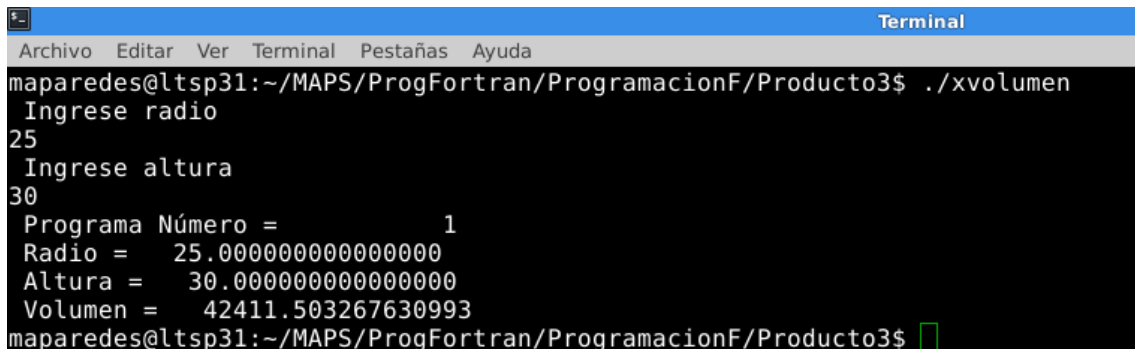
Este programa es muy similar al pasado en que se pide al usuario que ingrese datos. Lo que se obtiene con este programa el volumen de liquido contenido en un cuerpo .

```
!=====
! Volumen.f90 : Calcula el volumen de una esfera
!=====

program Volumen_Esfera !Inicio de programa

  Implicit None
  Real *8 :: radius , altura , vol , P3 !Declaracion de variables
  Real *8 :: PI = 4.0*atan(1.0)
  Integer :: model_n =1
  print *, 'Ingrese radio' !Hablar con el usuario
  read *, radius !Leer respuesta de usuario
  print *, 'Ingrese altura' !Hablar con el usuario
  read *, altura !Leer respuesta de usuario
  P3= 3.00 * radius - altura
  vol= 1.00 / 3.00 * PI * altura * altura * P3 !Calculo del volumen
  print *, 'Programa Número =', model_n !Muestra los resultados obtenidos
  print *, 'Radio =', radius
  print *, 'Altura =' , altura
  print *, 'Volumen =', vol

end program Volumen_Esfera
```



```
Terminal
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$ ./xvolumen
Ingrese radio
25
Ingrese altura
30
Programa Número =          1
Radio =  25.000000000000000
Altura =  30.000000000000000
Volumen =  42411.503267630993
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$
```

2.3. Programa: Precisión

Con este programa se puede apreciar que tan preciso son los cálculos que se obtienen. En este caso es un código de doble precisión.

```

!=====
! Precision.f90 : Determina la precision de la maquina
!=====
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 ! Recalcular one
    print *, i , one , epsilon_m ! Imprimir los valores
  end do ! End loop when i > n
end program Limits

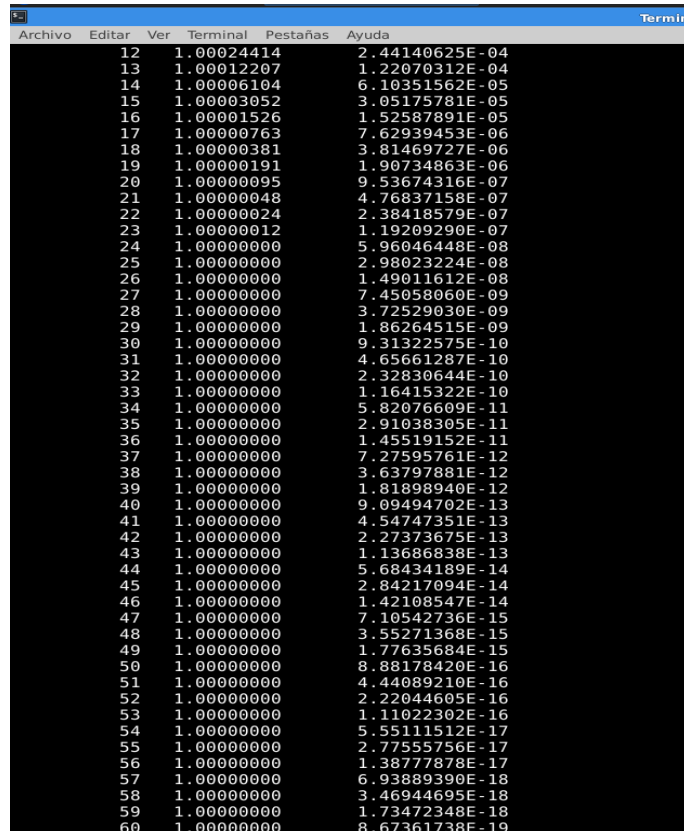
```

hitar	Ver	Terminal	Pestañas	Ayuda	Terminal
12		1.0002441406250000			2.4414062500000000E-004
13		1.0001220703125000			1.2207031250000000E-004
14		1.0000610351562500			6.1035156250000000E-005
15		1.0000305175781250			3.0517578125000000E-005
16		1.0000152587890625			1.5258789062500000E-005
17		1.0000076293945312			7.6293945312500000E-006
18		1.0000038146972656			3.8146972656250000E-006
19		1.0000019073486328			1.9073486328125000E-006
20		1.0000009536743164			9.5367431640625000E-007
21		1.0000004768371582			4.7683715820312500E-007
22		1.0000002384185791			2.3841857910156250E-007
23		1.0000001192092896			1.1920928955078125E-007
24		1.0000000596046448			5.9604644775390625E-008
25		1.0000000298023224			2.9802322387695312E-008
26		1.0000000149011612			1.4901161193847656E-008
27		1.0000000074505806			7.4505805969238281E-009
28		1.0000000037252903			3.7252902984619141E-009
29		1.0000000018626451			1.8626451492309570E-009
30		1.0000000009313226			9.3132257461547852E-010
31		1.0000000004656613			4.6566128730773926E-010
32		1.0000000002328306			2.3283064365386963E-010
33		1.0000000001164153			1.1641532182693481E-010
34		1.0000000000582077			5.8207660913467407E-011
35		1.0000000000291038			2.9103830456733704E-011
36		1.0000000000145519			1.4551915228366852E-011
37		1.0000000000072760			7.2759576141834259E-012
38		1.0000000000036380			3.6379788070917130E-012
39		1.0000000000018190			1.8189894035458565E-012
40		1.0000000000009095			9.0949470177292824E-013
41		1.0000000000004547			4.5474735088646412E-013
42		1.0000000000002274			2.2737367544323206E-013
43		1.0000000000001137			1.1368683772161603E-013
44		1.0000000000000568			5.6843418860808015E-014
45		1.0000000000000284			2.8421709430404007E-014
46		1.0000000000000142			1.4210854715202004E-014
47		1.0000000000000071			7.1054273576010019E-015
48		1.0000000000000036			3.5527136788005009E-015
49		1.0000000000000018			1.7763568394002505E-015
50		1.0000000000000009			8.8817841970012523E-016
51		1.0000000000000004			4.4408920985006262E-016
52		1.0000000000000002			2.2204460492503131E-016
53		1.0000000000000001			1.1102230246251565E-016
54		1.0000000000000000			5.5511151231257827E-017
55		1.0000000000000000			2.7755575615628914E-017
56		1.0000000000000000			1.3877787807814457E-017
57		1.0000000000000000			6.9388939039072284E-018
58		1.0000000000000000			3.4694469519536142E-018
59		1.0000000000000000			1.7347234759768071E-018
60		1.0000000000000000			8.6736173798840355E-019

2.4. Programa: Precisión 2

Mismo programa que el anterior con la diferencia de que los cálculos son menos precisos (menos decimales) debido a la forma en que declararon los variables.

```
!=====
! Precision2.f90 : Determina la precision de la maquina
!=====
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 ! Recalcular one
    print *, i , one , epsilon_m ! Imprimir los valores
  end do ! End loop when i > n
end program
```



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
48	1.00000000	3.55271368E-15
49	1.00000000	1.77635684E-15
50	1.00000000	8.88178420E-16
51	1.00000000	4.44089210E-16
52	1.00000000	2.22044605E-16
53	1.00000000	1.11022302E-16
54	1.00000000	5.55111512E-17
55	1.00000000	2.77555756E-17
56	1.00000000	1.38777878E-17
57	1.00000000	6.93889390E-18
58	1.00000000	3.46944695E-18
59	1.00000000	1.73472348E-18
60	1.00000000	8.67361738E-19

2.5. Programa: Math

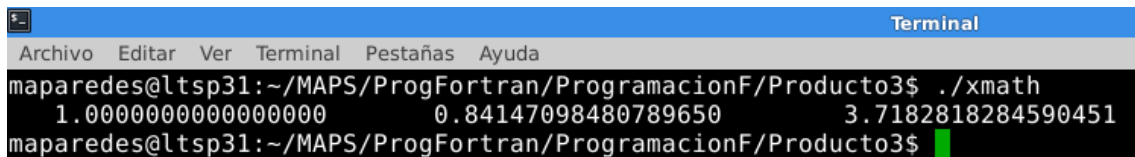
El lenguaje Fortran maneja funciones trigonométricas y las especiales. Lo que se realizó fue evaluar diferentes funciones e imprimir el resultado.

```
!=====
! Math.f90 : Desmostracion de funciones matematicas
!=====
```

```
program Math_Test

  Real *8 :: x= 1.0 , y , z
  y= sin (x)
  z= exp (x) + 1.0
  print *, x , y , z

end program Math_Test
```



```
Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$ ./xmath
1.0000000000000000 0.84147098480789650 3.7182818284590451
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$
```

2.6. Programa: Math 2

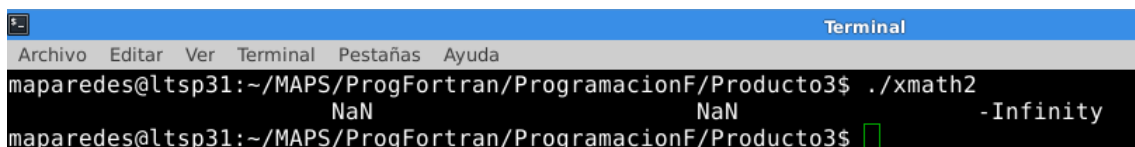
Este programa es muy parecido al anterior con la diferencia a que se evalúan otras funciones que tienen resultados que no son reales.

```
!=====
! Math2.f90 : Desmostracion de funciones matematicas
!=====
```

```
program Math_Test

  Real *8 :: x= -1 , y=2.0 , z=0
  Real *8 :: xx , yy , zz
  xx= sqrt (x)
  yy= acos (y)
  zz= log (z)
  print *, xx , yy , zz

end program Math_Test
```



```
Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$ ./xmath2
NaN NaN -Infinity
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$
```

2.7. Programa: Función

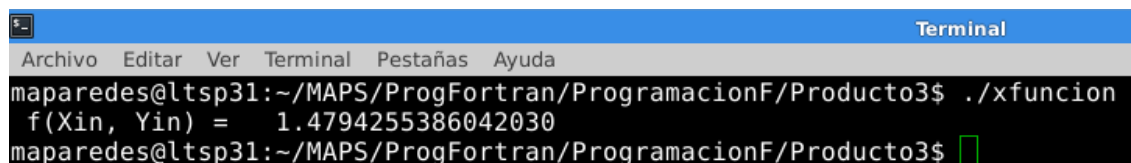
En este programa se declara una función fuera de lo que es el programa principal con el propósito de poder utilizarla esta función múltiples veces si tener la necesidad de declarar todo de nuevo.

```
!=====
! Funcion.f90 : Llamar a una funcion
!=====

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
  c = f ( Xin, Yin )
  write ( * , * ) 'f(Xin, Yin) =' , c

end Program Main
```



```
Terminal
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$ ./xfuncion
f(Xin, Yin) = 1.4794255386042030
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$
```

2.8. Programa: Subrutinas

Las subrutinas son pequeños programas que son llamados por el programa principal para realizar ciertas acciones, lo que permite que el trabajo sea mas limpio y fácil de manipular.

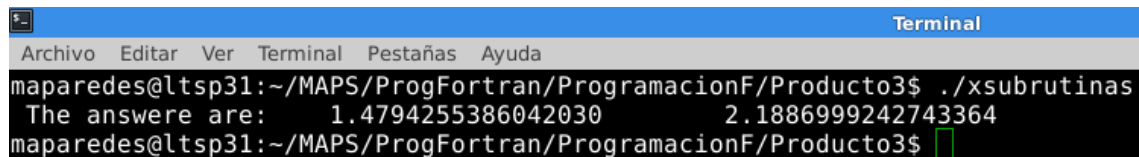
```
!=====
! Subrutinas.f90 : Demuestra el llamado de subrutinas
!=====
Subroutine g(x, y, ans1 , ans2 )

    Implicit None
    Real (8) :: x , y , ans1 , ans2
    ans1= sin (x*y) + 1
    ans2= ans1**2

end Subroutine g
!
program main_program

    Implicit None
    Real *8 :: Xin=0.25 , Yin=2.0 , Gout1 , Gout2
    call g(Xin, Yin, Gout1, Gout2 )
    write ( * , *) 'The answe are: ' , Gout1 , Gout2

end program main_program
```



```
Terminal
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$ ./xsubrutinas
The answe are: 1.4794255386042030 2.1886999242743364
maparedes@ltsp31:~/MAPS/ProgFortran/ProgramacionF/Producto3$
```