

1. Haz un programa en F90 que resuelva la ecuación de segundo grado, donde a, b y c son los coeficientes de la ecuación y se consideran como números reales solamente, toma en cuenta los casos cuando  $a=0$ , si es así el programa deberá poner el letrero en este caso "la ecuación es lineal y la raíz es :" y dar el resultado, si  $a=0$  y  $b=0$  poner el letrero "no hay solución", considerarás también los casos cuando el discriminante es igual a cero, mayor que cero y menor que cero, si es cero el programa deberá escribir en pantalla "la raíz es doble" y deberá obtener la raíz, si es mayor a cero deberá obtener las dos raíces y si el discriminante es menor que cero, poner el letrero "no hay raíces reales" y obtener las soluciones complejas.

```
PROGRAM Chicharronera
IMPLICIT NONE
REAL :: a, b, c, det, x_1, x_2
COMPLEX :: x_1_cmplx, x_2_cmplx

! Este programa recibe como parametros 3 números reales de precisión
! sencilla que representan los coeficientes de una ecuación cuadrática,
! tambien se usan otras variables del mismo tipo y precisión como auxiliares
! para presentar las raíces de dicho polinomio

WRITE (*,*) "Ingresa el coeficiente cuadrático"
READ (*,*) a

WRITE (*,*) "Ingresa el coeficiente lineal"
READ (*,*) b

WRITE (*,*) "Ingresa el término independiente"
READ (*,*) c

det = b**2 - 4*a*c !variable auxiliar

IF ((a /= 0) ) THEN !sirve para comprobar que sea una ecuación de segundo grado

IF (det > 0) THEN!raices reales

x_1 = (-b + sqrt(det)) / (2*a) !raiz 1
x_2 = (-b - sqrt(det)) / (2*a) !raiz 2

WRITE (*,*) "Las raíces del polinomio"
WRITE (*,*) a, "x**2 +", b, "x +", c
WRITE (*,*) "son:"
WRITE (*,*) "-----"
WRITE (*,*) x_1, x_2

ELSE IF (det < 0) THEN!raices complejas

x_1_cmplx = cmplx(-b/(2*a), sqrt(-det) / (2*a)) !raiz 1
```

---

```

x_2_cmplx = cmplx(-b/(2*a),-sqrt(-det)/(2*a)) !raiz 2

WRITE (*,*) "Las raíces del polinomio"
WRITE (*,*) a, "x**2 +", b, "x +", c
WRITE (*,*) "son:"
WRITE (*,*) "-----"
WRITE (*,*) x_1_cmplx, x_2_cmplx

ELSE IF (det == 0) THEN!raiz doble

x_1 = -b / (2*a)

WRITE (*,*) "La raíz es doble"
WRITE (*,*) "La raíz del polinomio"
WRITE (*,*) a, "x**2 +", b, "x +", c
WRITE (*,*) "es:"
WRITE (*,*) "-----"
WRITE (*,*) x_1

END IF

ELSE IF (a == 0 .AND. b /= 0) THEN!lineal

x_1 = -c/b

WRITE (*,*) "La ecuación es lineal y la raíz es:", x_1

ELSE IF (a == 0 .AND. b == 0) THEN!sin solución
WRITE (*,*) "No hay solución"

END IF

END PROGRAM Chicharronera

```

Los resultados obtenidos coinciden con los resultados arrojados por Wolfram, así que debe estar bien

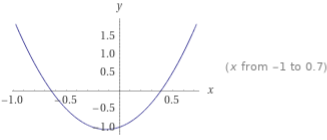
```
misael@misael-VirtualBox:~/Descargas$ gfortran poli.f90
misael@misael-VirtualBox:~/Descargas$ ./a.out
Ingresa el coeficiente cuadrático
4
Ingresa el coeficiente lineal
1
Ingresa el término independiente
-1
Las raíces del polinomio
4.00000000 x**2 + 1.00000000 x + -1.00000000
son:
-----
0.390388191 -0.640388191
misael@misael-VirtualBox:~/Descargas$ ./a.out
Ingresa el coeficiente cuadrático
1
Ingresa el coeficiente lineal
3
Ingresa el término independiente
9
Las raíces del polinomio
1.00000000 x**2 + 3.00000000 x + 9.00000000
son:
-----
(-1.50000000,2.59807611) (-1.50000000,-2.59807611)
misael@misael-VirtualBox:~/Descargas$
```

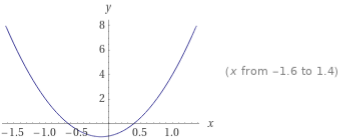
```
misael@misael-VirtualBox:~/Descargas$ ./a.out
Ingresa el coeficiente cuadrático
0
Ingresa el coeficiente lineal
4
Ingresa el término independiente
-1
La ecuación es lineal y la raíz es: 0.250000000
misael@misael-VirtualBox:~/Descargas$ ./a.out
Ingresa el coeficiente cuadrático
0
Ingresa el coeficiente lineal
0
Ingresa el término independiente
1
No hay solución
misael@misael-VirtualBox:~/Descargas$
```

Input:

$4x^2 + x - 1$

Plots:





Geometric figure: Properties

parabola

Alternate forms: More

$x(4x + 1) - 1$

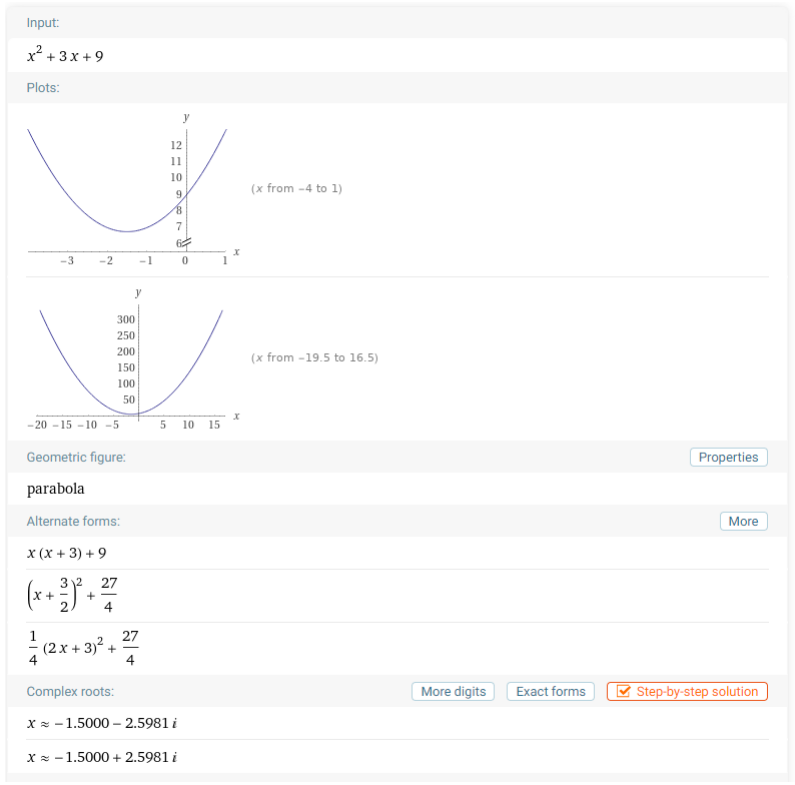
$4\left(x + \frac{1}{8}\right)^2 - \frac{17}{16}$

$\frac{1}{16}(8x + 1)^2 - \frac{17}{16}$

Roots: More digits Exact forms Step-by-step solution

$x \approx -0.64039$

$x \approx 0.39039$



2. Haz un programa en F90 que resuelva la serie de Taylor del coseno, hasta un número dado de términos y un argumento x, los cuales son la entrada del programa,

$$\cos x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$

considera que para cambiar el signo en los términos, en f90 puede usarse:  $(-1)^i$  donde i es un contador, deberás usar dos “do’s anidados” (uno para llevar la suma y otro para el factorial). Compara los resultados de tu programa con los obtenidos por la función intrínseca cos(x).

```
program taylor
implicit none
real, parameter :: pi = 3.14159265359
integer, parameter :: extra = selected_real_kind(p=24,r=1000)
integer :: i, j, m, k, y, z
real(extra), dimension(2000) :: fact, coss
real(extra) :: x, xx, realcos, scos = 1
!serie de taylor alrededor del 0 (serie de Mclaurin) para aproximar el coseno
!en cualquier valor gracias a la traslación se que usa

write (*,*) "¿Cual es el valor que quieres evaluar?(en radianes)"
read (*,*) x

y = floor(x/pi)
z = modulo(y,2)

if (abs(x) .GT. 2*pi) then !traslación para evitar que algún dato de la serie exp
xx = x - ((y-z)*pi)

else if (abs(x) .LE.2* pi) then
xx = x
end if

fact(1) = 1

do i = 2, 2000, 1 !calculo de los factoriales necesarios
fact(i) = fact(i-1) * (i)
end do

do i = 1, 1000, 1 !elementos de la serie

coss(i) = ((-1)**i) * ((xx**(2*i))/(fact(2*i)))
end do

do i = 1, 1000, 1 !suma de todos los elementos
```

```

scos = scos + coss(i)
end do

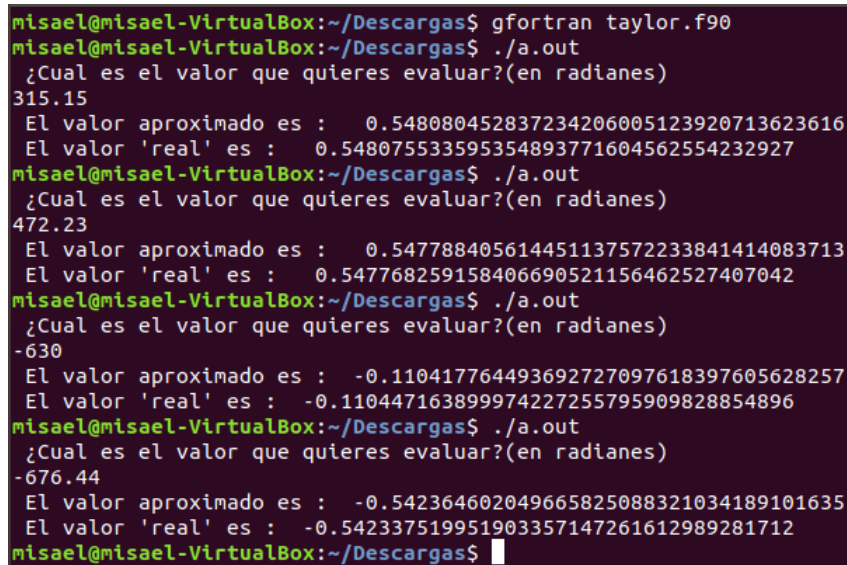
realcos = cos(x) !coseno "real"

write (*,*)"El valor aproximado es : ", scos
write (*,*)"El valor 'real' es : ", realcos

end program taylor

```

Los valores aproximados son muy parecidos a los obtenidos con la función intrínseca tanto para valores grandes y negativos



```

misael@misael-VirtualBox:~/Descargas$ gfortran taylor.f90
misael@misael-VirtualBox:~/Descargas$ ./a.out
¿Cual es el valor que quieres evaluar?(en radianes)
315.15
El valor aproximado es : 0.548080452837234206005123920713623616
El valor 'real' es : 0.548075533595354893771604562554232927
misael@misael-VirtualBox:~/Descargas$ ./a.out
¿Cual es el valor que quieres evaluar?(en radianes)
472.23
El valor aproximado es : 0.547788405614451137572233841414083713
El valor 'real' es : 0.547768259158406690521156462527407042
misael@misael-VirtualBox:~/Descargas$ ./a.out
¿Cual es el valor que quieres evaluar?(en radianes)
-630
El valor aproximado es : -0.110417764493692727097618397605628257
El valor 'real' es : -0.110447163899974227255795909828854896
misael@misael-VirtualBox:~/Descargas$ ./a.out
¿Cual es el valor que quieres evaluar?(en radianes)
-676.44
El valor aproximado es : -0.542364602049665825088321034189101635
El valor 'real' es : -0.542337519951903357147261612989281712
misael@misael-VirtualBox:~/Descargas$

```

3. Haz un programa en F90 que dado un numero  $n > 2$  obtenga todos los primos anteriores.

```

program primos
implicit none
integer :: n, i, k, cuenta
!programa que muestra los numeros primos anteriores al ingresado

write (*,*)"ingresa un entero mayor a 2"
read (*,*) n

if (n .GT. 2) then

do i = n, 2, -1
cuenta = 0
do k = i, 2, -1
if (modulo(i,k) .EQ. 0) then
cuenta = cuenta+1

```

```

end if
end do
if (cuenta .EQ. 1) then
write (*,*) i
end if
end do

else

write (*,*) "no es un numero mayor a 2"

end if

end program primos

```

Los primeros 100 números primos que nos da el programa son los mismo que dice Wolfram, por lo que debe estar bien

```

ingresa un entero mayor a 2
100
97
89
83
79
73
71
67
61
59
53
47
43
41
37
31
29
23
19
17
13
11
7
5
3
2
misael@misael-VirtualBox:~/Descargas$

```

Input interpretation:

primes
less than or equal to 100

Values:
Less

2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 | 31 | 37 | 41 | 43 | 47 | 53 | 59 | 61 | 67 | 71 | 73 | 79 | 83 | 89 | 97 (25 primes)

4. Haz un programa en F90 que realice mínimos cuadrados lineales con los datos obtenidos de un archivo con < puedes usar el imports-85.data cortando las columnas 13 y 26. Compara tus resultados con los obtenidos en gnuplot.

```
program min

character(10) :: a(205,26)
integer :: i,j
real :: x(205), y(205)

do i = 1, 205
  read (*,*) (a(i,j), j= 1, 26)
end do

do i = 1, 205
  read(a(i,13), *) x(i)
  read(a(i,26), *) y(i)
end do

print*, "b = ", (sum(y)*sum(x*x)-sum(x)*sum(x*y))/(205*sum(x*x)-sum(x)*sum(x))
print*, "m = ", (205*sum(x*y)-sum(x)*sum(y))/(205*sum(x*x)-sum(x)*sum(x))

end program min
```

El programa es el mismo que el de clase y la gráfica parece ajustarse bien a los datos, así que igual debe estar bien

```
misael@misael-VirtualBox:~/Descargas$ ./a.out < imports-85.data
b = -15199.8359
m = 523.946655
```

```
misael@misael-VirtualBox:~/Descargas$ gnuplot

      G N U P L O T
      Version 5.2 patchlevel 2      last modified 2017-11-01

      Copyright (C) 1986-1993, 1998, 2004, 2007-2017
      Thomas Williams, Colin Kelley and many others

      gnuplot home:      http://www.gnuplot.info
      faq, bugs, etc:    type "help FAQ"
      immediate help:    type "help" (plot window: hit 'h')

terminal type is now 'qt'
gnuplot> p "imports-85.data" u 13:26
gnuplot> b = -15199.8359
gnuplot> m = 523.946655
gnuplot> p "imports-85.data" u 13:26, x*m+b
```



