Computación - Práctica 6

Programación en FORTRAN:
Funciones.

Ejercicio 1. Supongamos que en el programa principal tenemos,

```
      INTEGER MAXINT
      PARAMETER (MAXINT = 32767)
      REAL X, Y, Z, FCAGLP
      INTEGER M, N
```

y una función `FCAGLP`, declarada como

```
      REAL FUNCTION FCAGLP(A, B, X)
      REAL A, B
      REAL X
```

¿Cuáles de las siguientes referencias a la función *fcaglp* son incorrectas en el programa principal y por qué?

a) `FCAGLP(A, B, X)`

b) `FCAGLP(FCAGLP, Y, M)`

c) `FCAGLP(Y, Z, N)`

d) `FCAGLP(3.5, 4.5, 6)`

e) `FCAGLP(X, Y, M)`

f) `FCAGLP(X, Y)`

g) `FCAGLP(Z, X, MAXINT)`

h) `FCAGLP(3.5, 4.5, 7.2)`

Ejercicio 2. *a*) Escribir una función de sentencia *conv(c)* para convertir temperaturas dadas en grados Celsius (°C) a temperaturas dadas en grados Farenheit (°F), donde $F = 1.8\,C + 32$.

b) Escribir un programa que genere una tabla de conversión de °C a °F, para °C = 1, 2, 3, ..., 100.

Ejercicio 3. En el plano una rotación de ángulo θ alrededor del origen transforma las coordenadas (x, y) de un punto en nuevas coordenadas (x', y') dadas por

$$\begin{cases} x' = x\,\cos\theta + y\,\mathrm{sen}\,\theta, \\ y' = -x\,\mathrm{sen}\,\theta + y\,\cos\theta. \end{cases}$$

Implementar un programa donde el usuario ingrese las coordenadas de un punto y el ángulo de rotación y obtenga las nuevas coordenadas utilizando funciones externas para realizar la transformación.

Ejercicio 4. En un programa hemos declarado una matriz real *A(20,20)*. Leemos de un archivo una matriz $A \in \mathbb{R}^{N\times N}$, con $N < 20$. Deseamos ahora calcular la traza de esta matriz, es decir, queremos hacer $T_r = \sum_{j=1}^{N} A_{jj}$; para lo cual definiremos una función *Tr*. De las siguientes posibilidades, ¿cuál producirá el resultado correcto, y por qué? (en el segundo caso, *np=20* al llamar a la función).

```
REAL FUNCTION TR(A, N)
REAL A(N,N)
INTEGER N, J
TR = 0.
DO J = 1, N
   TR = TR + A(J,J)
ENDDO
RETURN
END
```

```
REAL FUNCTION TR(A, NP, N)
REAL A(NP,NP)
INTEGER N, J, NP
TR = 0.
DO J = 1, N
   TR = TR + A(J,J)
ENDDO
RETURN
END
```

Verificar escribiendo un programa con las dos funciones dadas, (llamarlas TR1 y TR2) y usando la matriz contenida en el archivo *P06-Matriz.dat*.

Ejercicio 5. Hacer una función que calcule $n!$ y con ella obtener

$$C(n,r) = \frac{n!}{r!\,(n-r)!},$$

Ejercicio 6. Dado un vector complejo de N elementos, escribir una función que devuelva el elemento de mayor o menor módulo, dependiendo del valor de uno de sus argumentos, que debe ser de tipo carácter.

Ejercicio 7. Escribir un programa que, para una matriz real $A^{m\times n}$, nos permita calcular su *norma de Frobenius* $\|A\|_{frob} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$, o su *norma infinita* $\|A\|_\infty = \max\limits_{i,j}\{|a_{ij}|\}$. Usar una función para cada caso; la elección de la norma debe ser dejada al usuario al momento de correr el programa; usar una variable tipo carácter para identificar esta opción. Usar la matriz del ejercicio 4 para verificar el programa.

Ejercicio 8. El *máximo común divisor* (MCD) de dos enteros positivos J y K es un entero con la propiedad de dividir a ambos J y K (con resto nulo) y es el mayor de todos los divisores comunes. El algoritmo de Euclides para hallarlo es:

1. Sea J el mayor entero, y K el menor.
2. Sea R el resto de J dividido por K.
3. Mientras $R \neq 0$

   Hacer $J = K$

Hacer $K = R$

Hacer R el resto de J dividido por K.

4.Imprimir el valor de K como el MCD.

Generar una función para hallar el MCD entre dos enteros positivos y escribir un programa que llame a esta función; probar con varios ejemplos su correcto funcionamiento.

Ejercicio 9. El método de la Regula Falsi permite calcular los ceros o raíces de una función mediante la fórmula:

$$x_{n+1} = \frac{f(x_n) * x_{n-1} - f(x_{n-1}) * x_n}{f(x_n) - f(x_{n-1})}$$

siendo x_{n+1} la raíz en la n-ésima iteración.

Escribir un programa para encontrar las raíces de $f(x) = \cosh(x) - 2x$, con un error relativo (ver fórmula en el ejercicio 9 de la práctica 4) menor que 10^{-4}. La función $f(x)$ deberá estar definida mediante una función de sentencia.

Hallar los dos puntos iniciales de arranque graficando con gnuplot. Imprimir, con formato, el valor de la raíz y la cantidad de iteraciones en cada caso.

Ejercicio 10. El método de Simpson para calcular integrales definidas de la forma $\int_a^b f(x)dx$ consiste en dividir el intervalo $[a, b]$ en n subintervalos (con n par) de longitud h y aproximar la integral mediante la fórmula:

$$\int_a^b f(x)dx \approx \frac{h}{3} * (E + 4I + 2P)$$

donde:

$E = f(a) + f(b)$ es la suma de los valores de $f(x)$ evaluada en los extremos del intervalo,

$I = f(x_1) + f(x_3) + ... + f(x_{n-1})$ es la suma de los valores impares de $f(x)$,

$P = f(x_2) + f(x_4) + ... + f(x_{n-2})$ es la suma de los valores pares de $f(x)$,

con $x_i = a + i\,h$.

Calcular la integral $\int_0^1 e^{-x^2}\,dx$, escribiendo dos versiones del programa: uno que utilice un subprograma *FUNCTION* para el cálculo de la integral y, dentro del mismo, una función de sentencia para el integrando y otro que utilice dos subprogramas *FUNCTION*.

Imprimir en pantalla el resultado, la longitud h y la cantidad de puntos usados. Verificar, que a medida que se aumenta la última cantidad, el resultado tiende al valor "exacto" 0.746824.

Los que tengan deseos de "jugar" un rato, pueden calcular también $\int_a^b p(x)dx$, donde $p(x)$ es un polinomio de primer o segundo grado; tanto los límites de integración como los coeficientes del polinomio pueden ser elegidos arbitrariamente. Comparar el resultado con la integral calculada analíticamente.

Ejercicio 11. (Opcional) Utilizando la función del ejercicio 5 para calcular el factorial de un número, construir un subprograma *FUNCTION* para obtener la combinatoria $C(n, r)$. Para n no muy grandes, la función factorial se hace muy grande, con lo cual si utilizamos la expresión del ejercicio 5 el programa nos dará resultados erróneos. En lugar de ello, usar la siguiente expresión

$$C(n, r) = \frac{n\,(n-1)\,(n-2)\cdots(n-r+1)}{r!} = \frac{n\,(n-1)\,(n-2)\cdots(n-r+1)}{1\cdot 2\cdots(r-1)\,r}$$

Utilice esta función para calcular las N primeras filas del Triángulo de Tartaglia o Pascal.