Capítulo 1

Estructuras de Control - Do While()

1.1. Do While()

La sentencia **Do While()** es la combinación entre el **DO** y el **IF()** y básicamente es hacer un ciclo de sentencias que se repiten mientras una pregunta sea siempre **VERDADERA**. Se escribe de la siguiente manera:

DO WHILE (expresión lógica)
  sentencia 1
  sentencia 2
  sentencia 3
  sentencia 4
ENDDO

Ventajas:

- Es muy útil cuando se tiene en claro una pregunta que responder, la más típica de estas es: ¿Llegué con los cálculos que hace el programa que está corriendo a un error menor que cierto número? Por ejemplo, tomemos el caso de una serie de Taylor, donde puedo tener un estimador del error. Si como resultado de esa pregunta, si se ha llegado a un error por el cual no se necesita seguir calculando, rompo el ciclo que calcula la serie de Taylor y por lo tanto, el programa continua con las órdenes siguientes. Pero por otro lado, si no arribé al error óptimo se siguen calculando términos de la serie.

- Cuando los incrementos en las variables no son aditivos (sumas o restas) sino fórmulas más complejas.

- Los programas son compactos y fáciles de leer.

Detalles a tener en cuenta:

- Las variables que son necesarias modificar (incrementar o decrementar) no son controladas por el **Do While()**, si no que se lo debe programar. Por ejemplo en una serie de término i, debo agregar la sentencia i = i + 1 para que en cada bucle se incremente la variable que me permite calcular el próximo término de esta serie. A diferencia del la sentencia **Do**, donde el comienzo, final y paso están claramente indicados, en el **Do While()** esto no sucede. El programador debe verificar que en cada bucle se produzca una evolución de las variables que permita llegar a un final del loop.

- Un error del programador en la construcción de la pregunta puede hacer que el programa quede trabado en el bucle y la corrida en esta situación no terminaría nunca. Por ejemplo, podría pasar que la serie que se está calculando no converja, con lo cual nunca se podría volverse falsa la pregunta del **Do While()** y entonces no se saldría del loop que se programó.

1.1.1. Ejemplos

Kepler en 1609 publicó las leyes que rigen el movimiento de los planetas. Estos giran alrededor del Sol en una órbita elíptica, uno de cuyos focos lo ocupa el Sol, pero no lo hacen con un movimiento uniforme, sino que el radio vector Sol-planeta barre áreas iguales en tiempos iguales. La expresión matemática de esta ley para este movimiento proyectado sobre una circunferencia equivalente es la ecuación de Kepler:

$$M = E - e\ sen(E)$$

donde:

M es la anomalía media o ángulo que recorrería un planeta ficticio que se moviese con movimiento uniforme si la órbita fuese una circunferencia cuyo diámetro coincide con el eje principal de la elipse. E es la anomalía excéntrica (el ángulo real del planeta sobre su órbita elíptica y e (tal que $0 \leq e < 1$) es la excentricidad de la elipse.
M y e se tienen en tablas, pero la posición real del planeta es E, que es el valor que debemos encontrar.
despejando:

$$E = M + e\ sen(E)$$

El proceso para resolver esta ecuación trascendente se llama iteración[1]. De tal manera que comienzo con un valor arbitrario de E (lo llamo E0). Con ese valor y la ecuación anterior, calculo un nuevo valor de E (lo llamo E1). Este E1 es ahora mi nuevo E0 para calcular un nuevo valor E1. Si el sistema converge, la diferencia $|E1 - E0|$ se irá achicando en cada ciclo y me servirá como cota máxima del error. De esta manera usando un Do While() puedo repetir el proceso todas las veces que sea necesario hasta que el valor obtenido tenga un error menor al valor necesitado.

Veamos un programa que realiza la iteración usando un **Do While()**

```
Program kepler
real*8 M, E0, E1, exc

PI=3.1415926
write(*,*) 'Ingrese el valor de la excentricidad'
read(*,*) exc
write(*,*) 'Ingrese el valor de la Anomalía Media'
read(*,*) M

M = M/180*PI
```

[1] Los fundamentes teóricos de porque una iteración funciona en un caso o en otro no es un tema del Análisis Numérico y existen maneras de asegurar la convergencia a una solución con este método

```
E0=0
E1=1
DO WHILE(abs(E1-E0).gt.1e-8)
  E0 = E1
  E1 = M + exc * sin(E0)
  write(*,*) E0,E1,E0-E1
ENDDO

 write(*,*) 'La Anomalía Excéntrica es:',E1/pi*180,' Grados'

 end
```

Si compilo y corro el programa obtengo:

Ingrese el valor de la excentricidad
0.09
Ingrese el valor de la Anomalía Media
23.9874
1.0000000000000000 0.49439150927584963 0.50560849072415037
0.49439150927584963 0.46136377059387018 3.3027738681979446E-002
0.46136377059387018 0.45872439398428766 2.6393766095825222E-003
0.45872439398428766 0.45851154689480583 2.1284708948182685E-004
0.45851154689480583 0.45849437016298522 1.7176731820611746E-005
0.45849437016298522 0.45849298392430504 1.3862386801788418E-006
0.45849298392430504 0.45849287204816341 1.1187614162855297E-007
0.45849287204816341 0.45849286301921632 9.0289470899840296E-009
La Anomalía excéntrica es: 26.269705256849726 Grados

Note que la tercera columna es la cota máxima del error y que esta disminuye muy rápidamente asegurando un muy buen resultado y en poco tiempo.

Ejemplo: Vayamos a un problema que ya hemos resuelto, el cálculo la serie de Taylor de la función cos(x), con un error menor a error $< 10^{-6}$

$$Cos(x) = \sum_{n=1}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ... + error(\xi)$$

Veamos el programa

El programa es simple, vamos calculando los términos de la serie y sumándolos, pero al mismo tiempo calculamos el término de error. Evaluando ese término sabemos si llegamos al error que necesitamos o no. Si no llegamos calculamos un término más a la serie y continuamos el proceso.

```
Program cos_de_angulo
```

```
      write(*,*) 'ingrese el valor del ángulo n'
      read(*,*) omega
      pi=3.14159265358979
      omega=omega/180*pi
      xmax=pi/4
      coseno=0.
      eterm=1
      i=0

      do while(eterm.gt.1e-8)

C       Calculo el factorial para el término y para del error: n! y (n+2)!
        facto=1.
        do ii=1,2*i
          facto=facto*ii
        enddo
        facto2=facto*(i+1)*(i+2)

C       Calculo el termino i
        term=omega**(2*i)/facto*(-1)**(i)

C       Calculo el término y el error
        coseno=coseno+term
        eterm= xmax**(2*i+2)/facto2

C       Incremento i para calcular el término que sigue
        i = i +1

      enddo

99      write(*,*) i,coseno,cos(omega)

      end
```