Computación

Importante

La división entre dos enteros da como resultado un entero (en realidad, cualquier operación entre enteros da un entero)

—> Pierdo los decimales!!! En la división

Las operaciones entre clases diferentes de números se realiza en el campo de la variable de mayor jerarquía (la que necesita más recursos).

Los cálculos con enteros dan como resultado enteros.

Problemas con la precisión de los cálculos

a = 0.1 <- sistema decimal

b = 0.2 <- sistema decimal

Pero en binario

$0.1_{10} = 0.00011001100110011001100....$

$0.2_{10} = 0.00110011001100110011...$

Estructuras de control
Sentencia DO

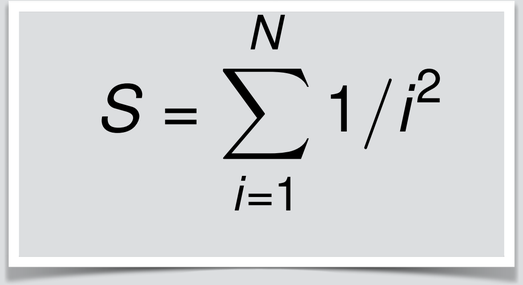
Sentencia DO

La sentencia DO se utiliza de esta manera:

Do *variable=inicio, final, paso*

…

ENDDO



Por ejemplo:

```
do i=1,100,1
    x=x+ i
enddo
```

Es decir la variable de control tiene un inicio, un final y un paso y repite las sentencias que están dentro de la sentencia DO. No importa cuántas sentencias puedan ser.

La variable de control del Do puede ser REAL pero es conveniente en lo posible usar números enteros. Dentro del DO la variable de control sólo puede ser modificada por el propio DO.

El paso si es 1 puede no figurar.

La variable no puede ser modificada por una orden que no sea DO dentro de este.

Puede haber un DO dentro del otro, pero el más interno tiene obligatoriamente que cerrar antes que el externo.

El DO interno que manejar otra variable

Si no inicio el paso, este es 1.

Ejemplo: hagamos un ejemplo, un programa que sume la serie

$$S = \sum_{i=1}^{N} 1/i^2$$

C Programa para realizar el calculo de la suma de la

C Serie finita $1/i^2$

$$S = \sum_{i=1}^{N} 1/i^2$$

```
Program Suma
write(*,*) 'Cuántos términos quiero sumar?'
read(*,*) N
suma=0.
DO i=1, N
      x=i
      suma = suma + 1/ x**2       !¿otra solución?
ENDDO

write(*,*) 'La suma de la serie es =', suma
end
```

Veamos el programa en una terminal

Consideraciones

- La cantidad de términos de la serie es N, entonces el código sirve para distintos valores de N sin tener que editarlo y compilarlo.
- El tiempo es proporcional con $t \propto N$, si aumento N el tiempo que tarda el programa crece en forma lineal.
- El programa podría tardar mucho para un N muy grande.
- Al final del DO, i tendrá el valor i+1, pero el Do no se habrá ejecutado para ese valor.

La serie puede sumarse al revés

Es decir:

…

```
DO i=N,1,-1
   x=i
   suma = suma + 1/x**2
ENDDO
```

…

Puedo tener Do´s dentro de otros (anidados/nested)

```
Do I=1,n
    Do j=1,n
        … muchos cálculos
    Enddo
Enddo
```

Consideraciones:

La sentencia es ideal para representar sumatorias, productorias, operaciones con matrices, etc.

En caso de las sumatorias por ejemplo, una serie de Taylor.

Veamos el caso de un factorial

…

```
F=1

DO I=1,n

    F = F * I

ENDDO
```

Caso de DO`s Anidados

Realicemos el siguiente cálculo de la tabla F, tal que

$F = 2N+M$

Donde N toma valores tal que:

N = 2,4,6,8…,20.

Y la variable M los toma tal que:

M=1,2,3,4,…,N

Veamos como se realizaría el programa:

```
Program Tabla

integer F

write(*,*) ' N M F'
DO n=2,20,2
    DO m=1,n
        F= 2*n+m
        write(*,*) n,m,F
    ENDDO

ENDDO

END
```

Apéndice:

Como cambiar el tipo de número

De flotante o real a entero se usa float()

a= float(i)

De flotante a entero int()

i = int(a) <— Pierdo los decimales (trunco la parte decimal del número)

De flotante a complejo se usa complex()

Complex z

z=complex(x,y) <— Donde x es un flotante con la parte real, y es un flotante con la parte imaginaria