Hasta ahora tenemos una manera de introducir/sacar cantidades de un programa

* La entrada ha sido desde teclado
* La salida ha sido a pantalla

Existen muchas situaciones donde el ingreso de datos desde teclado se vuelve impráctico.

Podemos escribir un archivo en donde se encontrarán todos los datos que el programa necesita para generar una solución

En Fortran tenemos la instrucción:

OPEN(lista de argumentos)

**Creación y características de nuestro archivo:**

La lista de argumentos es importante. Por ahora solo se considerara seis de los principales argumentos (ayudan a controlar el archivo de texto: abrirlo, escribir en él, etc etc)

Si el archivo no existe Fortran lo crea usando los argumentos que le damos como base.

1. **UNIT** - Este argumento especifica la unidad del archivo. La unidad es una constante de tipo INTEGER que se asocia a un archivo particular
2. **FILE** - indica el nombre del archivo que se va a utilizar. Debe de ser una constante de tipo CHARACTER
3. **STATUS** - indica el estatus del archivo. Debe ser una constante de tipo CHARACTER.

Estatus disponibles:

1. *NEW* - el archivo no existe antes de la ejecución del programa pero se crea durante la misma. Cualquier intento de asignar el estatus nuevo a un archivo ya existente termina en un error de compilación.
2. *OLD* - el archivo existe antes de la ejecución del programa y este lo utilice. Error de compilación si usas el estado OLD pero el archivo no existe previamente.
3. *UNKNOWN* - Estatus desconocido. Si el archivo existe lo usa y si no, lo crea. (combinación de old y new ¿?)
4. *SCRATCH* - permite construir archivos de texto temporales. El archivo se genera durante la ejecución y se elimina al terminarlo. Si se utiliza este estatus, el argumento FILE debe omitirse, en caso contrario se tendrá un error de compilación.
5. **ACTION** - en este argumento se indica la acción a realizar en el archivo de texto. Requiere un argumento de tipo CHARACTER.

a) READ - archivo de solo lectura

b) WRITE - permite escribir en el archivo

c) READWRITE - se puede leer y escribir en el.

1. **IOSTAT** - EN este argumento se almacena un número de tipo INTEGER asociado a un posible error en el OPEN. Nos indica en dónde está el error que cometimos. Su uso evita el rompimiento del programa ante un posible error.

Cuando se usa IOSTAT en el READ se tienen 3 posibilidades:

1. No hay problema alguno (asigna un valor de 0)
2. Ocurre un problema (asigna un valor entero positivo)
3. Si el archivo se termina (asigna un valor entero negativo)
4. **IOMSG** - Es una variable de tipo CHARACTER que nos muestra un mensaje que nos indica el error cometido.

**EJEMPLO**

OPEN(UNIT = 3, FILE =”noods”, STATUS = “NEW”, ACTION=”WRITE”, IOSTAT = error, IOMSG = mensaje\_error)

WRITE(3,\*) “Hola a todos”

WRITE(3,\*) “Bienvenidos al curso de nuevo”

CLOSE(3)

Ya no se puede escribir en este archivo hasta que se vuelva a abrir

Variables usadas en el ejemplo

INTEGER :: error

CHARACTER(80) :: mensaje\_error

La cantidad de caracteres necesario para mostrar el mensaje completo

Ejemplo de escritura en un archivo de texto

PROGRAM nombre

IMPLICIT NONE

INTEGER :: num\_error

CHARACTER(len=80) :: mensaje\_error, nombre\_archivo

PRINT\*, "Introduce el nombre del archivo."

READ\*, nombre\_archivo

OPEN(UNIT = 325, FILE = nombre\_archivo, STATUS = "OLD", ACTION = "WRITE", IOSTAT = num\_error, IOMSG = mensaje\_error)

**! La unidad puede tener cualquier valor, solo debe de ser entero y positivo**

**! Por si se detecta un error, y qué hacer cuando se detecte**

IF(num\_error == 0) THEN

! Si no hay error, IOSTAT almacena un cero en la variable

WRITE(325,\*) "Weno Dia"

WRITE(326,\*) "Welcome to Hell, nibba"

ELSE

PRINT\*, "Se detectó el error numero", num\_error

PRINT\*, mensaje\_error

END IF

CLOSE(325)

END PROGRAM texto

**! Se detecta un problema por que se supone que el archivo ya existe pero no se puede encontrar.**

Se puede cambiar el estatus a NEW para crear un archivo nuevo en donde aparezcan los mensajes

* Construir un programa que permita escribir un archivo para su posterior graficación

f(x) = 3.2 cos(x\*\*2)

Se debe de pedir un intervalo a-b. El conjunto de x debe iniciar en a y terminar en b.

El intervalo se va a partir en n pedazos de la misma longitud. El usuario decide el valor de n

Para determinar cada x, haremos uso de la misma técnica que usamos en el caso del programa del trapecio

h = b-a/n

xi = a + i\*h

se generan los puntos que queremos evaluar

El valor de la función se calcula con cada xi

Habrá una tabla con dos columnas(x,y) en donde se pone un valor de x y se calcula la función.

**Para graficar en terminal: gnuplot**

plot “nombre\_archivo”, función utilizada

plot “graf”, 3.2\*cos(x\*\*2)

muestra la función graficada (se muestra una línea continua)

splot - graficar en 3d

para mostrar los nombres de los ejes

set xlabel “eje x”

set ylabel “eje y”

replot

se usa cuando hay una modificación en la gráfica

para exportar la gráfica como imagen

set term png

set output “nombre.png”

replot

quit (cerrar gnuplot)”