Capítulo 1

Entrada y Salida de datos de un código

Este capítulo es sólo para dar una vista rápida de los modos básicos de las sentencias que manejan los sistemas de lectura y escritura de datos. Esto es con respecto al ingreso de datos por teclado, su lectura de una unidad de almacenamiento (disco rígido, SSD, Pen drive, etc), adquiridos de un dispositivo conectado a la computadora o que se decida guardar datos en forma permanente. Lo que normalmente serían las órdenes de entrada/salida (Input/Output en inglés o con sus iniciales I/O de datos de un programa)
Más adelante en un capítulo especial sobre este tema veremos estos comandos con mucho más detalle (Cap. 9)

1.1. Lectura

Para la lectura de datos se utiliza la sentencia READ (leer en inglés). Esta orden en su versión más simple, necesita muy pocos parámetros para que realice su trabajo. Hay que determinar de dónde se lee, como se lee y que se lee. La sentencia en su forma mínima podría escribirse asi:

READ(*,*) A,B,C,D

El primer "*" es la **Unida Lógica** normalmente es un número que indica de donde se lee, si tengo un "*" es que se leerá del teclado en el lugar donde se está corriendo el programa. Es decir, de ahí se leerán los números que se escriban. En el caso del ejemplo, tengo que leer 4 variables, así que la computadora esperará que se escriban 4 números separados por al menos un blanco (barra espaciadora), terminando el ingreso al apretar la tecla de retorno (return o enter según el teclado).

El segundo "*" indica como se lee, es decir la cantidad de decimales. Lo habitual es que en ese lugar se escriba un número, y que ese número indique donde se encuentra una sentencia que se llama FORMAT en la cual se puede definir la cantidad de lugares que ocupa el número o si hay que saltear espacios, o bien que tipo de número estoy leyendo (entero, real, etc). Mas adelante, discutiremos esta sentencia y sus comandos. En este caso particular que no hay un número y está el "*" con lo cual se cede la decisión a la computadora para realizar este trabajo. Por lo cual la cual leerá todos los dígitos decimales encuentre para cada número y considerará que los números son separados por blancos. Si hubiese más números que variables para leer los números restantes se ignorarán. si se escriben menos números y se apreta la tecla de entrada, el programa seguirá esperando que se completen la cantidad de números restantes para que cada variable tenga un valor ingresado.

1.2. Escritura

La sentencia WRITE realiza la función contraria del READ. Con ella podemos escribir el resultado de un programa en el sistema de almacenamiento, en la pantalla o quizás en una impresora. Se usa de una forma similar a la sentencia READ:

WRITE(*,*) X,Y,Z

El primer "*" de la sentencia **READ**, es la **Unida Lógica** y es una indicación del lugar donde escribo, normalmente un número[1]. Este número identifica un archivo, la impresora, mi monitor, etc. Si hay un "*" o un "1" el lugar donde se escribirán las variables es la pantalla de donde se está corriendo el programa. En este caso, se imprimirán los números que están guardados en las variables X,Y,Z separadas por un espacio.

El segundo "*" es el formato (FORMAT) y funciona en forma similar a como lo indicamos en la sección de la sentencia READ. En este caso se imprimirán todos los decimales de los números.

1.3. Archivos

Los archivos (files en inglés) son la unidad de almacenamiento en los discos rígidos y demás sistemas que guardan información en forma permanente. Los archivos se localizan en directorios cuya función es la organización de la información. En cada directorio sólo puede existir un solo archivo con un nombre determinado, es decir no puede repetirse el mismo nombre en otros archivos para un dado directorio, pero si puede estar un archivo con nombre similar en otros directorios. En la mayoría de los sistemas operativos, los directorios pueden contener otros directorios y estos a su vez más directorios.
Las reglas sobre los nombres y tamaños de los archivos están dadas por el **File System** que es la definición de cómo se formateó el disco (ver capitulo 1). Hay que recordar que los Sistemas Operativos, pueden manejar varios tipos de File Systems, por ejemplo en la actualidad la mayoría de los pen-drives vienen formateados de fábrica con el File System FAT32 y aunque es un sistema de formateo de la empresa Microsoft, tanto el Linux, como el MacOSX (Apple) lo pueden leer. Incluso las máquinas fotos suelen usar este formato que es uno de los más comunes en las tarjetas de memoria.
Para leer o escribir un archivo secuencial (el tipo más usado) se usa la sentencia **OPEN()**.

OPEN(22, fiile='estrellas.dat')

En este caso estoy asignando el archivo que se llama estrella.dat a la unidad lógica número 22. Entonces podría leer de ese archivo con la sentencia:

READ(22,*) X1,X2,X3

Es decir leo un renglón y los tres número que leo los asigno a las variables X1,X2,X3
Pero también puedo escribir sobre otro archivo haciendo:

OPEN(35, fiile=´salida.txt´)
WRITE(35,*) A,B

[1] Por ejemplo, en muchos sistemas la impresora es la Unidad Lógica Nº 6.

Donde aquí escribo sobre el archivo *salida.txt*