Capítulo 1

Introducción a Fortran

El nombre Fortran viene del inglés “The IBM Mathematical **For**mula **Tran**slating System”, es decir sistema traductor de fórmulas de IBM. Con el tiempo acortaron el nombre a FORTRAN. Es un lenguaje de alto nivel[1] diseñado para ser utilizado en los ambientes científicos, básicamente de cálculo. Las primeras versiones del lenguaje son del año 1957 y de ahí en adelante ha evolucionado siendo la base de muchos lenguajes modernos. Puede manejar variables, incluyendo vectores y matrices. Tiene muchas décadas de uso, testeo y optimización, por lo cual es muy eficiente ya que sobrellevó muchas correcciones que lo mejoraron durante un largo período de tiempo (más de cuatro décadas).

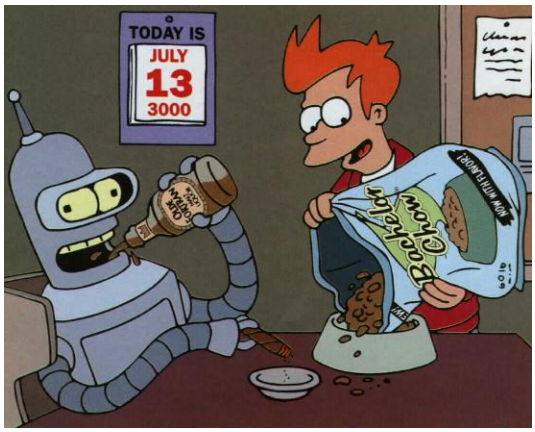

Figura 1.1. Bender en la serie Futurama tomando Olde Fortran en el año 3000

Tiene puntos a favor y también puntos en contra. Si resumimos, estos son:

A favor:

- Mucha experiencia y compiladores extremadamente eficientes. Son muy rápidos y con resultados finales excelentes.
- Es un lenguaje simple, muy rápido de aprender. Pocas órdenes y concisas frente a lenguajes más modernos.
- Compatible en sus estructuras con otros lenguajes (C o Python).
- Los compiladores pueden optimizar el código para el hardware existente.
- Existen Infinidad de algoritmos ya programados, que se pueden encontrar en libros o en internet.
- Se puede escribir en mayúscula o minúscula indistintamente.
- Se aprende rápido, y el código escrito de un programa es fácil de leer.

En contra:

[1] Existen lenguajes que se llaman de bajo nivel, en los cuales se programa a nivel de ordenes de la CPU.

- Alguna de la sintaxis de las ordenes provienen de la época que se perforaban tarjetas.
- No es orientado a objetos.
- En los lenguajes modernos hay muchas más funciones y algoritmos pre-programados (bibliotecas).
- No incluye un sistema para hacer dibujos o gráficos.
- No es interactivo y no tiene mucha utilidad fuera de los requerimientos de científicos o de ingeniería.

Aunque parece que hay otras consideraciones sobre el Fortran fuera de este planeta (ver figura 1.2).

1.1. Asignaciones

Si doy en Fortran la siguiente orden:

I=5
IMA=23
FE4=484.22

Estoy guardando el número que está a la derecha del signo "=" en el nombre (imagínense que es una caja) que está a la izquierda. Es decir en la variable que se llama "I" guardo dentro de ella el número 5 (un número entero), en la que se llama IMA guardo un 23 (entero). Pero en la que se llama FE4 guardo un número real el 484.22.
Esta operación se conoce con el nombre de **asignación** y en este caso el símbolo de "=" causa la asignación, pero este "=" no es para indicar una igualdad, es decir, no es el "igual" que acostumbramos ver en una ecuación, aunque como veremos más adelante se le parece mucho. Los nombres de la variables siempre deben empezar con una letra, pero después de esa letra pueden tener números. Otros lenguajes (ni Fortran, ni Python) para evitar la confusión entre ecuaciones y asignaciones han usado otro símbolo para la operación de asignación, pero en la mayoría se usa el "=".

Figura 1.2. ¡Fortran!

Hay que prestar atención que las computadoras (al igual que las calculadoras de mano) utilizan la nomenclatura anglosajona para los puntos y las comas, es decir las usan al revés que en español. La coma que usamos para indicar la parte fraccionaria del número es ahora un punto. Es decir, el número 23,5 (23 y medio) es ahora 23.5 (con punto en vez de coma).

1.2. Constantes y variables

Hay 5 tipos de variables en Fortran distintas y con propiedades únicas. Tres de estos tipos guardan números. Estas son:

Enteras (Integer)
Reales (Float)
Complejas (Complex)

Lógicas (Logical/Boolean)
Texto (Character)

Veamos cada una de ellas:

1.2.1. Variables Enteras

Las variables Enteras en Fortran, pueden ser de 2, 4 u 8 Bytes, y se definen usando la sentencia (u orden) **Integer*n** donde n es la cantidad de Bytes que voy usar en ese número en particular. Con más Bytes se pueden escribir números con mayor cantidad de dígitos tanto negativos como positivos). Si el nombre de la variable empieza con alguna de estas letras I,J,K,L,M o N la variable será entera por definición (default)[2], a menos que se de una orden en contrario. Las variables enteras son muy útiles para manejar los subíndices en un matriz o en un vector, por ejemplo.
Si escribo:

INTEGER cuenta

La variable **cuenta**, ahora sólo sirve para cargar números enteros, no podría albergar a un número real. Por ejemplo, si ahora ordeno la siguiente operación:

cuenta = 3.141549

En cuenta solo se habrá guardado un 3 y los decimales se perdieron. Ya que cuenta es un número entero y estos no pueden guardar decimales.

1.2.2. Variables Reales

Las variables reales en Fortran son de 4 Bytes (**REAL*4**) o 8 Bytes (**REAL*8** o también conocidas como doble precisión). Cualquier variable con un nombre que comience con las letras de A hasta la H y desde la O hasta la Z, es por definición **REAL*4**. Si quiero que sea **REAL*8** tengo que indicarlo con un orden específica que será:

REAL*8 Mag, Mag1, Mag2

Habiendo dado esta orden las variables Mag, Mag1, Mag2 ahora sólo guardan números reales de 8 Bytes, mientras que si no hubiese dado esa orden serían variables enteras porque sus nombres comienzan con M.
Existe la orden **DOUBLE PRECISION** y es equivalente en todo a poner **REAL*8**.
En casos muy extremos puede usarse la orden REAL*16 (precisión cuádruple) donde la representación binaria el número será mucho más extensa (128 bits) permitiendo una menor pérdida de precisión en la operaciones, pero en la mayoría de las computadoras no es soportada por el hardware, y por lo tanto es una orden que se ejecuta corriendo software, por lo cual no sólo es importante el consumo de memoria ram (4 veces más que un REAL*4) sino que el tiempo de cálculo se vuelve mucho más largo. Dicho de otra manera, utilizar variables REAL*16 implica entender que se usarán muchos más recursos en la computadora tanto en tiempo como en memoria y sólo debe utilizarse en casos que lo justifiquen.

[2] En inglés: default, con esta palabra se indican las definiciones predeterminadas que ya se han adoptado, incluso en un sentido más amplio que sólo el tipo de variable.

1.2.3. Variables Complejas

Las variables complejas utilizan dos números, uno para la parte real y otro para la parte imaginaria. Los números complejos pueden ser real*4 o real*8, y definido un número como complejo ambas parte real e imaginaria son del mismo tipo. Los complejos en Fortran se representan como un par ordenado (a,b), en donde a es la parte real y b la imaginaria. (a,b) es el número a + b i.
Para construir una variable para que guarde números complejos tengo que usar la orden **complex**. Ejemplo:

```
COMPLEX A1,A2,A3,A4
```

De aquí en adelante las variables A1,A2,A3 y A4 sólo guardarán números complejos.

1.2.4. Lógicas

Las variables lógicas o Booleanas, solo pueden contener un Verdadero (en inglés:True) o Falso (False). Para ellos usan un Byte, en el que sólo activan o no uno de bits (si, desperdiciando 7 bits). En Fortran, el Verdadero se escribe como **.true.** y el false como **.false.**. Note que ambas palabras tienen **puntos** adelante y atrás. Estos puntos se ponen para diferenciar un verdadero o falso de una variable o texto que se llamara “true” o “false”.
Para que una variable sea Booleana tengo que usar la orden **Logical**. Ejemplo:

```
LOGICAL L1,L2,L3
```

Ahora las variables L1, L2 y L3 sólo guardarán Verdaderos o Falsos.

1.2.5. Variables de caracteres

Las variables de caracteres, utilizan 1 Byte por cada letra que yo quiera representar. Por ejemplo si quiero guardar un texto de 25 caracteres, tendría que definir la variable para esta cantidad de caracteres o más (me pueden sobrar, aunque eso agregaría espacios en blanco). Lo haría de esta manera:

```
CHARACTER*25 cartel
```

Entonces la variable cartel guarda un texto de hasta de 25 caracteres. La idea de que un Byte es un caracter viene de la definición de códigos ASCII que veremos más adelante, en ella cada letra tiene un número binario de un byte definido por convención para todos los fabricantes de computadoras, esa definición fue modificada para albergar mas caracteres en los que se llama UNICODE donde se llegan a usar hasta 4 Bytes.

1.3. Vectores, matrices y cubos

En Fortran hay una manera de convertir cualquiera de los tipos de variables que vimos antes, en un vector, matriz, cubos y estructuras con más dimensiones. La orden más antigua para esta tarea es la DIMENSION, y se usa así:

```
DIMENSION A(10)
```

Esta orden que se pone al principio del programa, define que la variable A, tiene 10 componentes y que estas serían: A(1),A(2),A(3),...,A(9) y A(10). Cada una de estas componentes actúa ahora como una variable que puede guardar números. Si escribiera la siguiente sentencia:

DIMENSION B(100,100)

Ahora B es una matriz de 100x100 elementos, y por ejemplo existe como variable B(22,97)
La forma más moderna de usar esta orden es así:

REAL*8 B(100,100), C(100,100,100)
donde aprovecho y fijo el tipo de variable y su dimensión.

1.4. RESUMEN

Las variables que comiencen su nombre con una letra determinada son reales (real*4) o enteras. Si no me sirve esa definición, la puedo forzar con un comando. Por ejemplo, en astronomía medimos los flujos de energía que emite una estrella como su magnitud. Magnitud empieza com M entonces la palabra sólo guardaría un número entero, pero eso no nos sirve, las magnitudes son números reales, entonces, defino:

real*4 magnitud

Ahora con esta nueva definición me sirve para guardar números reales.
Puedo re-definir de una sola vez y en una sentencia, muchas variables variables:

real*8 ixag, jxag1, kxag2, lxag4
integer xa, xb, xf5, ser
complex*8 a, b, c
complex*8 x(100) $\rightarrow$ X es ahora un vector de números complejos y doble precisión de 100 elementos

1.5. Definición de Variables en Fortran 90/95

En Fortran 90/95 se cambió la forma de definir las variables. En este nuevo sistema se separan con "::"(dos :) la parte del tipo de variable, de una lista de nombres de variables que se definirán de ese tipo específico.

```
Tipo específico:: Lista de Variables
```

Veamos unos ejemplos:
Las variables ZIP, Media and Total quiero que sean del tipo INTEGER

```
INTEGER :: ZIP, Media, Total
```

Las variables promedio, error, sum and ZAP quiero que sean del tipo REAL

```
REAL::promedio, error, sum, ZAP
```

Y las de tipo CHARACTER

```
CHARACTER(LEN=15) :: Name, Street
```

LEN=15 significa que se usan 15 lugares (Bytes) para las letras