Capítulo 1

Asignaciones y Funciones Intrínsecas

1.1. Asignaciones

Como hemos contado, las órdenes en Fortran se escriben como una serie de renglones como un texto. Cada uno de estos renglones es una orden (o sentencia) que se ejecutará una detrás de la otra en orden. El primer renglón es la primera orden, luego de esa se ejecuta la sentencia del segundo renglón y así sucesivamente hasta la último (aunque existen órdenes que permiten volver a pasar por las mismas sentencias una y otra vez que estudiaremos en los siguientes capítulos).
Ya hemos adelantado lo que es una asignación en la sección anterior a esta, es decir, cómo cargar una constante en una variable del programa. Por ejemplo:

I=5

donde a la variable entera I le asignamos el número 5.

1.2. Operaciones

En Fortran se utilizan las operaciones básicas con los símbolos que utilizamos comúnmente (+, -, /), salvo por el signo de multiplicación donde no usa la “x” sino que se usa el símbolo “*”, que de hecho muchos teclados de computadora ya la tienen marcada así, en la zona derecha en la parte del teclado numérico. Otra operación básica que tiene una notación distinta es la potencia. No podemos poner x^2 porque no puedo escribir en mitad de los renglones de una computadora (no hay supraíndices, tampoco subíndices). Por eso la potencia se escribe como dos asteriscos seguidos. Es decir, x^2 se escribe como $x**2$
Las prioridades de las operaciones básicas son las mismas que las que se establecen en el álgebra. Por ejemplo, puedo escribir:

A = b1 + c4 / x

Esta orden dividiría el valor guardado en c4 por x y el resultado de ese valor se lo sumaría a b1. El resultado final de toda la operación se guardaría en A. Note que las multiplicaciones y divisiones tienen prioridad sobre las sumas y restas. Pero puedo usar los paréntesis adecuados para acomodar el cálculo a mi gusto. Por ejemplo:

A = (b1 + c4) / x

Ahora las operaciones se hacen en diferente orden, por la posición de los paréntesis se sumarían b1 más c4 y recién el resultado se dividiría por x.

Las asignaciones no son ecuaciones, y veamos por qué. ¿Cómo haría si quisiera saber en un cierto código cuántas veces este programa pasa por un determinado lugar, para volver a realizar el mismo cálculo? La idea sería tomar una variable para usar como contador, y sumarle un uno cada vez que paso por el lugar donde se encuentra. Es decir con una sentencia como esta:

I = I + 1

En esta sentencia se busca el valor de I que se encuentra en la memoria, se le suma un uno y se lo vuelve a guardar en la variable I. Como vemos esto no es una ecuación, es un procedimiento de derecha a izquierda donde se guarda el resultado. Por eso las asignaciones no son ecuaciones pero se les parecen, nada evita que yo escriba:

E = M * C2**

En donde hago el cálculo de energía de la famosa ecuación de Einstein $E = MC^2$. Pero no hay que confundirse, una es una ecuación de la física y la otra es la sentencia para hacer el cálculo en Fortran.

1.2.1. Funciones Intrínsecas

El lenguaje Fortran tiene una serie muy importante de funciones matemáticas preprogramadas, estas incluyen trigonometría, raíz cuadrada, logaritmos, exponencial, etc. Muchas de estas funciones se calculan usando el hardware, es decir, hay circuitos en la CPU que las pueden calcular a velocidades muy altas. En la tabla 1.1 veremos algunas de las más usadas.

Función	Nombre en Fortran	Función	Nombre en Fortran
sen(x)	SIN(X)	cos(x)	COS(X)
tan(x)	TAN(X)	arcsen(x)	ASIN(X)
arccos(x)	ACOS(X)	arctan(x)	ATAN(X)
ángulo(y, x)	ATAN2(Y,X)*	$\mid x \mid$	ABS(X)
$\sqrt{x}$	SQRT(X)	$e^{(x)}$	EXP(X)
ln(x)	LOG(X)	log(x)	LOG10(X)

Tabla 1.1. Algunas funciones en Fortran
* Da el ángulo con su cuadrante a partir de las coordenadas del par ordenado (X,Y)
Note que el logaritmo natural se escribe como el de base 10, LOG(X),
y el de base 10 en Fortran se escribe como LOG10(X)
Los argumentos de las funciones trigonométricas son siempre en radianes, no en grados

Ejemplos:
$z = \sqrt{(x^2 + y^2)}$
en Fortran sería: **z = sqrt(x**2 + y**2)**

$x = e^{\frac{1}{4}y^2}$

en Fortran sería: **x = exp(1/4*y**2)**

$z = \frac{1+\frac{1}{x}}{3x+2}$

en Fortran sería: **z= (1+1/x)/(3*x+2)**

$\omega = \cos(\alpha + \phi) + \cos\alpha\cos\phi - \text{sen}\ \alpha\ \text{sen}\ \phi$

en Fortran sería: **omega = cos(alfa + fi) + cos(alfa) * cos(fi) - sin(alfa) * sin(fi)**
donde al no tener letras griegas, escribo sus nombres, como nombres de las variables que uso.

1.3. Variables vectoriales

Una vez que, por ejemplo, la variable A está definida como un vector de 10 elementos, podríamos hacer lo siguiente:

```
 REAL*8 A(10)

 A(8) =22.543434
I = 5
A(9)= A(8) * A(I)
```

En este caso, como **I** toma el valor 5, el A(I), es A(5). Es decir, los índices de los vectores y matrices pueden también ser variables, que por razones obvias tienen que ser enteras. Por lo cual, si se acuerdan de los teoremas del Álgebra sobre matrices, estos siempre se definen sobre un elemento genérico $a_{(i,j)}$ y en Fortran sería la variable **A(I,J)**. Dicho en otras palabras, en Fortran y los demás lenguajes de computación podemos manejar el mismo nivel de abstracción que en matemática.

1.4. Estructura de un programa Fortran

En esta sección analizaremos la estructura básica de un programa Fortran (que es similar a otros lenguajes). Primero hay que ver las reglas para escribir los renglones y recordar que cada renglón es una orden.
Estas son:
En Fortran 77 (y anteriores), las primeras 6 posiciones se reservan y las órdenes se deben escribir a partir de la columna 7 hasta la columna 72. Si en las primeras columnas aparece una letra **C** o un *****(muy raro que vean este símbolo para esto), ese renglón es un comentario sin ninguna orden activa, lo cual es muy bueno para hacer anotaciones de lo que trata lo que estamos programando, y por ejemplo, qué significa cada variable o de dónde sacamos el algoritmo, etc. Siempre es bueno tener muchos comentarios sobre lo que se hace, para recordar datos útiles de la tarea que se realiza en ese segmento del programa. A veces se trabaja en un grupo de investigación con otras personas y buenos comentarios ayudan a una mejor interacción con los colaboradores. Fortran 90 agregó el símbolo **!** como otra indicación de comentario con la ventaja de que puede ser puesto en el mismo renglón que una sentencia activa a continuación de esta.

Si tengo una orden muy larga, y ya llegué escribiendo a la columna 72 y necesito más espacio para escribir, lo que tengo que hacer es incluir en la línea de abajo algún caracter en la columna 6 y eso le avisa al compilador que sigue la orden de la línea de arriba. Esto se puede repetir todo

lo que sea necesario, es decir, una sentencia podría extenderse por decenas de renglones. Se puede poner números de las columnas 0 a la 6 y esos números indican posiciones determinadas en el programa. Me permitirán hacer que mi programa retome alguna de esas líneas (veremos más adelante cómo hacer esto). Esos números que pueden ser discontinuados (se puede poner 99, sin que existan los 98 anteriores) actúan como si fuesen carteles indicando posición. Son sólo una etiqueta indicando un lugar en el programa.
Resumiendo:

Col. 1 En blanco o “c” o “*” para comentarios
Col. 1-5 : En blanco o uso como etiqueta (opcional)
Col. 6 : Continuación de la línea anterior (opcional)
Col. 7-72 : Sentencias
Col. 73-80: Se pueden usar como comentarios, ya que lo que está acá es ignorado por el compilador.

Ejemplo de continuación en la línea de que sigue abajo:

c23456789 (Uso este comentario para tener una referencia del número de columna!)
La siguiente sentencia la escribo en dos renglones:
area = 3.14159265358979
+ * r * r

Veamos un programa simple, pero completo en el sentido de que tiene todas las estructuras que se usan en programas mucho más grandes tanto en largo como en recursos. Para ello vamos a tener una meta, hacer un código que calcule el área de un triángulo, que es: Área = (Base x Altura) / 2.

Este sería el programa:

```
C234567
      Program areat

C    Programa para realizar el calculo del área de un triángulo rectángulo.
C    Ingresando la base y la altura}

      real*8 base, altura, area

      read(*,*) base, altura

      area = (base*altura)/2

      write(*,*) 'El area es =',area

      end
```

La idea es ahora analizar sentencia por sentencia lo que este programa hace y que significa frente a la estructura general que se utiliza para programar en Fortran.
Program areat

Esta orden da nombre al programa, puede tener algún significado especial en algunos SO o compiladores.

C Programa para realizar el calculo del área de un triángulo rectángulo.
C Ingresando la base y la altura
Estas dos líneas, empiezan con la letra C así que son comentarios, me sirven a mí, por ejemplo, para recordar que se está calculando, cuál es el significado físico de cada variable y cuál es el método del cálculo, etc. El compilador las ignora y para el resultado final da lo mismo que estén o no. Pero les recuerdo, es muy bueno comentar lo que se hace en cada sección de un programa.

real*8 base, altura, area

En esta orden convertimos las variables **base**, **altura** y **area** de real*4 que sería la definición estándar a real*8 que asegura más decimales, aunque hay que comentar que en este caso particular esto no tendría mucho sentido a menos que se justifique la necesidad de una mayor precisión en los cálculos. Esto es lo que se debe hacer al comienzo de los programas en Fortran (y en muchos otros lenguajes) tenemos que definir al comienzo la forma y el tipo de las variables que vamos a usar. Puede que en programa muy importante existan cientos de líneas definiendo variables.

read(*,*) base, altura

En esta sentencia hacemos una **entrada** de datos al programa, eso lo hace la orden **read** (leer en inglés). Esta orden tiene un paréntesis en el cual hay dos “*” . El primer “*” es de dónde yo leo, si está el “*” significa que el programa lee los números del teclado donde corre el programa. El segundo “*” es cómo los leo, ahí podría indicar por ejemplo la cantidad de decimales, etc. Si hay un “*” dejo que la computadora decida. Los asteriscos funcionan como una especie de definición estándar delegando en la computadora la toma de decisiones, en la mayoría de los casos puede ser una buena idea, pero no siempre. Luego en la orden están las dos variables a leer, por lo cual el programa detiene su ejecución y espera que escribamos en el teclado los valores. Primero uno y luego el otro separado por un blanco (también se podría haber puesto una coma separándolos).

area = (base*altura)/2

En esta sentencia hacemos el cálculo y asignamos el resultado a la variable **area**. Esta sería la zona de cálculo del programa, en otro programa ser el área de cálculo podría muy extensa y contar con miles de líneas.

write(*,*) ´El area es =´,area

En esta orden, hacemos lo contrario al **read**, ahora vamos a escribir el resultado guardado en la variable **area**, para ello usamos la orden **write** y en este caso el primer asterisco indica “donde estoy”, es decir mi pantalla y el segundo asterisco sin un formato, o sea todos los decimales. Lo que escribimos en la pantalla (o un printer) a continuación, y es el texto ´El area es =´ y luego el valor guardado en la variable **area**.
Por ejemplo, podríamos obtener como resultado el siguiente texto: El area es = 23.45566

La sentencia **end** es para avisar al compilador cuando crea el código ejecutable que el programa terminó.
Para compilar este programa, en linux escribimos: **gfortran triangulo.f -o triangulo** donde triangulo.f sería un archivo de texto que contuviese el programa que hemos analizado. El “-o” indica el nombre del programa ejecutable que se debe crear, en este caso “triángulo”.