Conceptos básicos de programación - I

Vamos a comenzar dos sesiones (8 horas) de introducción a la programación y para ello utilizamos el lenguaje de programación **Fortran**.[1] Estas sesiones están basadas en FORTRAN 77 que es el que aún se está utilizando en múltiples programas de química, sin embargo, también hemos añadido pequeñas actualizaciones a Fortran 90.

Lea detenidamente y con calma la introducción siguiente. Siempre que aparezca una **[C]** es que tiene que contestar alguna cuestión. Cuando aparezca una **[P]** es porque tiene que hacer un programa (editar, compilar y ejecutar) y cuando funcione imprimirlo. Los programas que no tengan la **[P]** no hace falta sacarlos por la impresora. Los programas impresos, las hojas del profesor y los ejercicios 1 a 3, forman el guión de prácticas que tiene que entregar al profesor cuando se le indique. También encontrará una serie de ejercicios adicionales que en caso de tener tiempo debe incluir en el guión de prácticas (OPCIONAL).

Ante un problema estudie detenidamente la información que le proporciona el ordenador e intente resolverlo por sí mismo antes de preguntar al profesor.

1. Editar, Compilar y Ejecutar

Lo primero que vamos a hacer es escribir, compilar y ejecutar nuestro primer programa en Fortran. Para ello siga los pasos siguientes:

1.- (EDITAR) Edite con cualquier editor un fichero llamado **prog1.f** con el texto de la Fig. 1. En esta figura el editor utilizado es el «kate», pero puede utilizar el que le resulte más cómodo (KWrite, vi, ...). Procure escribir el texto lo más literal posible respetando la columna en la que se escribe cada cosa. Salve (guarde) el fichero al disco con el nombre prog1.f

2.- (COMPILAR) [1] Ejecute la siguiente instrucción:

gfortran prog1.f -o prog1.exe (↩)

El símbolo final (↩) significa que debe pulsar dicha tecla o la tecla Intro para hacer operativa la orden. Si ha escrito bien el fichero prog1.f no debe aparecer ningún error. Si la compilación produce algún error, lea detenidamente lo que le indica el ordenador, y con esa información revise (reedite) el fichero prog1.f y vuelva a ejecutar la instrucción anterior. Si no consigue solucionar los problemas por sí mismo pregunte a su profesor.

3.- (EJECUTAR) Si la compilación no da ningún error ejecute el comando **ls** y verá que entre otros tiene en el disco los ficheros prog1.f y prog1.exe. Este último es un fichero ejecutable. Escriba la instrucción siguiente para ejecutar:

./prog1.exe (↩)

[1] Compilar consiste en traducir el lenguaje Fortran al código máquina que es capaz de entender el ordenador. Esta operación la realiza un programa llamado compilador, en este caso gfortran.[2] Si no funciona el comando gfortran consulte con su profesor la versión de compilador de Fortran instalada en el ordenador.

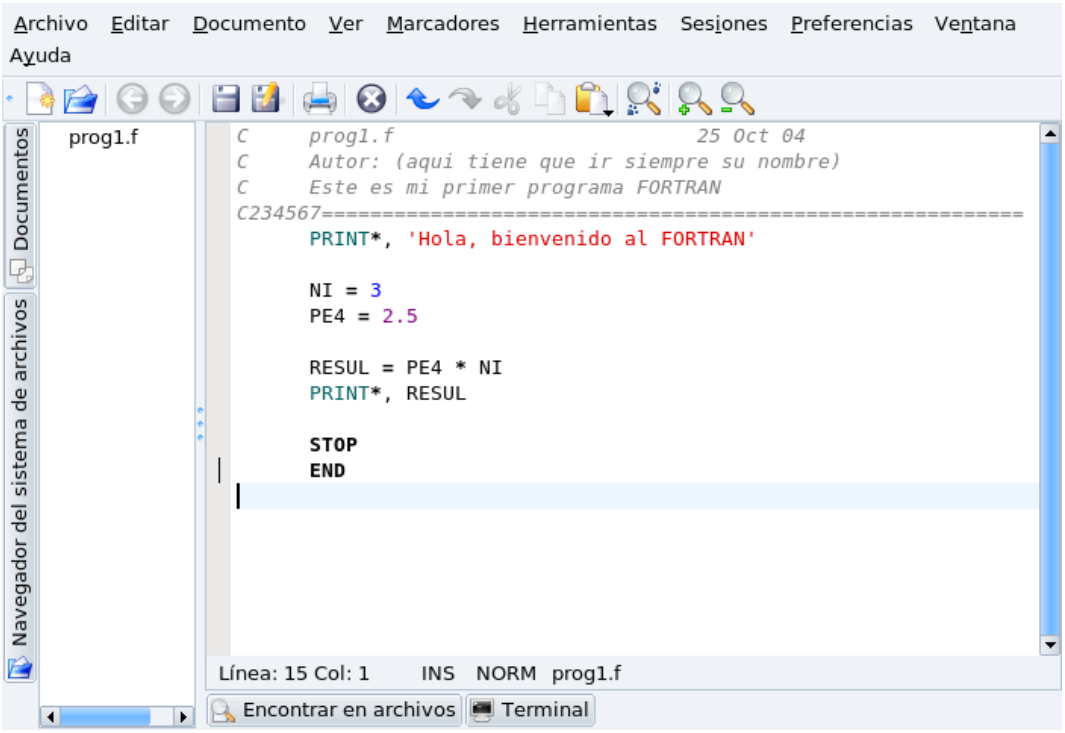


Figura 1: Primer programa Fortran (prog1.f) editado con el editor kate.

El «./» antes del nombre del ejecutable (prog1.exe) es necesario en las últimas versiones de linux e indica que el programa ejecutable esta en el subdirectorio de trabajo en el que estamos situados.

Si todo ha funcionado, entonces hemos editado, compilado y ejecutado nuestro primer programa en Fortran.

En estas dos sesiones de prácticas vamos a aprender un poco más sobre la programación Fortran. Como puede suponer, aquí estamos presentando una introducción muy breve a la programación utilizando el Fortran como punto de partida. Le aconsejamos que practique por su cuenta y utilice las referencias dadas en la bibliografía para profundizar en el tema.[3, 4, 5, 6]

[C] Anote en la hoja del profesor el resultado de este programa.

2. Constantes, Variables y Tipos de Datos

Fortran es un lenguaje de programación «vivo» ya que la mayoría de los programas usados por la comunidad científica están escritos en Fortran. Es un lenguaje de alto nivel, es decir, requiere poco conocimiento del hardware y emplea palabras conocidas (normalmente en inglés) que le indican al ordenador lo que tiene que hacer con unas serie de «objetos» con los que va a trabajar.

El tipo de «objetos» que va a manejar el ordenador siguiendo las órdenes de nuestro programa son los que se muestran en la Tabla 1. En estas prácticas únicamente vamos a trabajar con números enteros y reales y en raras ocasiones con caracteres. Por defecto, el Fortran que estamos usando utiliza 4 bytes ($8 \times 4 = 32$ bits) para guardar un número entero o un número real. Esto implica que el número entero más grande, en valor absoluto, que podemos usar es $2^{32-1} - 1 = 2147483647$. Por otro lado, los números reales tendrán una precisión aproximada de 7 cifras significativas y su rango estará entre 10^{-38} y 10^{+38}.

Tabla 1: Tipos de constantes y variables (datos) utilizados en el lenguaje Fortran.

Tipo	**FORTRAN 77**	**Fortran 90**	**Intervalo**
Enteros	INTEGER*1 *variable* INTEGER*2 *variable* INTEGER*4 *variable* (defecto)	INTEGER(1):: *variable* INTEGER(2):: *variable* INTEGER(4):: *variable*	$\pm$128 $\pm$32768 $\pm$2147483647
Reales	REAL*4 *variable* (defecto) REAL*8 *variable*	REAL(4):: *variable* REAL(8):: *variable*	$\pm$7 decimales, $10^{\pm 38}$ $\pm$16 decimales, $10^{\pm 308}$
Caracteres	CHARACTER**n variable* (1 byte por carácter)	CHARACTER(LEN=*n*):: *variable*	
Lógicos	LOGICAL *variable*	LOGICAL:: *variable*	
Complejos	COMPLEX *variable*	COMPLEX:: *variable*	

Los datos pueden estar en nuestro programa de dos maneras diferentes, como constantes o como variables (Fig. 2). Las variables son como las memorias de nuestra calculadora de bolsillo y su contenido puede cambiar durante la ejecución del programa.

Constantes: Cualquier número o carácter que aparece en una proposición.
Variables: Cualquier número o carácter al que se le asigna un nombre[a] y se le permite cambiar.

Ejemplos:
na25 = 7
xy = na25 + 12.5
nom = 'Error 1027.'

Constantes: 7, 12.5, 'Error 1027.'
Variables: na25, xy, nom

No son válidos nombres de variables como: 35n o $alpha, por ejemplo.

[a] Puede tener hasta 31 caracteres (letras o números). El primer carácter tiene que ser una letra.

Figura 2: Constantes y variables.

Los números enteros y reales se definen (es decir, se le indica al ordenador que son números enteros o reales) como se muestra en la Fig. 3. Por otro lado también tenemos varios tipos de números enteros y reales, aunque nosotros vamos a utilizar únicamente los que tiene el Fortran por defecto y que hemos señalado en la Tabla 1 (defecto).

Enteros:
- Las constantes no llevan punto decimal.
- Las variables comienzan por: i, j, k, l, m, n.
- Son palabras de 4 bytes (32 bits).
- El mayor «integer» es $2^{(N-1)} - 1$ donde N es el número de bits.

Reales:
- Las constantes deben llevar punto decimal.
- Las variables comienzan por una letra distinta de i, j, k, l, m, n.
- Son palabras de 4 bytes (32 bits).

Figura 3: Definición por **defecto** de enteros y reales.

Es importante recordar que **las variables enteras se definen con nombres que comienzan por i, j, k, l, m, n (i-n). Y variables reales son las que comienzan con las demás letras (a-h, o-z)**. Esta definición puede modificarse fácilmente pero para simplificar la programación vamos a mantenerla.

Con estos datos podemos hacer una serie de **operaciones básicas** (Fig. 4). Observe que **las operaciones tienen un orden de precedencia, es decir la multiplicación y la división siempre se realizan antes que las sumas y**

restas. Para evitar problemas a este respecto conviene utilizar paréntesis siempre que tengamos dudas, aunque no sea necesario.

- Tenemos las cuatro operaciones básicas: $+$, $-$, $*$ y $/$
- Y las potencias: $**$
- **El orden de precedencia de las operaciones es:** $** > *, / > +, -$
- Las instrucciones se ejecutar por orden de precedencia y de izquierda a derecha
- **Por ejemplo:** $i**j**k = (i**j)**k$
  $a/b/c = a/(b*c)$
  $(a+b)/(c+d) \neq a+b/c+d$

Figura 4: Operaciones numéricas básicas.

Una operación básica y a la vez difícil de entender cuando la vemos por primera vez es la **instrucción de asignación**. **Cuando vemos un signo = debemos entender que se realizan las operaciones que hay a la derecha del signo $=$ y el resultado se guarda en la variable que esta a la izquierda del igual**. Por tanto, a la izquierda del $=$ siempre debe haber una variable.

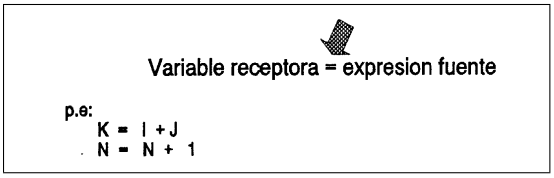


Figura 5: Instrucción de asignación.

Vamos a hacer un programa para repasar los conceptos vistos hasta ahora. EDITE, COMPILE Y EJECUTE el programa **prog2.f**. **Evidentemente, los números de la columna derecha (entre corchetes) sólo sirven para numerar las líneas y no hay que escribirlos.**

[C] DISCUTA brevemente los resultados y anótelos en la hoja del profesor.

```
C  prog2.f              16 Oct 97      { 1}
C  Autor: ....                         { 2}
C  conceptos basicos                   { 3}
C=====7========================        { 4}
      P1 = 5.3                         { 5}
      N1 = 3                           { 6}
      P2 = P1*N1  ! comentario ...     { 7}
      N2 = P1*N1                       { 8}
                                       { 9}
      PRINT*, '  P2: ', P2             {10}
      PRINT*, '  N2: ', N2             {11}
                                       {12}
      P2 = P2+N1 * 2                   {13}
      PRINT*, '  P2-NUEVO: ', P2       {14}
                                       {15}
      STOP                             {16}
      END                              {17}
```

En los programas en Fortran, **las instrucciones comienzan en la columna 7**. Los **comentarios** (las primeras cuatro líneas en los programas anteriores son comentarios) se indican con una C (también podría ser un asterisco * o el signo de cerrar admiración !) en la columna 1. Los comentarios son ignorados por el compilador. El programa termina con dos instrucciones que todo programa Fortran debe tener, **stop** y **end**.

El lenguaje FORTRAN77 (y anteriores) sigue una serie de reglas rigurosas en cuanto a la colocación de las instrucciones (estudie la Tabla 2 detenidamente).

Tabla 2: División de las líneas en Fortran.

FORTRAN 77	Fortran 90
Todas las líneas que llevan una **C** o un asterisco (*) en la 1^a columna son comentarios para el programador. El compilador ignora dichas líneas.	A partir del carácter cerrar admiración (**!**), el resto de la línea es un comentario ignorado por el compilador.
Las columnas 1-5 están dedicadas a las **etiquetas**.	
Las instrucciones Fortran van entre las columnas 7 a 72 con disposición libre (los espacios en blanco son ignorados).	La instrucción puede extenderse desde la columna 1 hasta la columna 132.
Cualquier carácter en la columna 6 indica que esta línea es **continuación** de la línea anterior	Para continuar una línea se debe incluir al final de la misma el carácter **&**.

[C] REEMPLACE la línea 13 del programa prog2.f por:

1. **P2 = (P2 + N1) + 1**
2. **P2 = N1 / N2**
3. **P2 = REAL(N1) / REAL(N2)**
4. **P2 = P2 / 2.0 * N1**
5. **P2 = P2 / (2.0 * N1)**

[C] Vuelva a COMPILAR y EJECUTAR cada caso. ANOTE en la hoja del profesor el resultado. Mantenga el mismo nombre (prog2.f) para todos los casos.

3. Funciones Intrínsecas o Predefinidas

Las funciones intrínsecas son aquellas definidas por el propio lenguaje de programación (por ejemplo el coseno, la raíz cuadrada, el logaritmo, etc). Están incluidas en las librerías del compilador y éste las reconoce de modo automático (ver Tabla 3). A continuación mostramos un programa sencillo que emplea funciones intrínsecas. Observa que los argumentos (valores o variables sobre los que se aplica la función) van siempre entre paréntesis y, si hay más de uno, separados por comas.

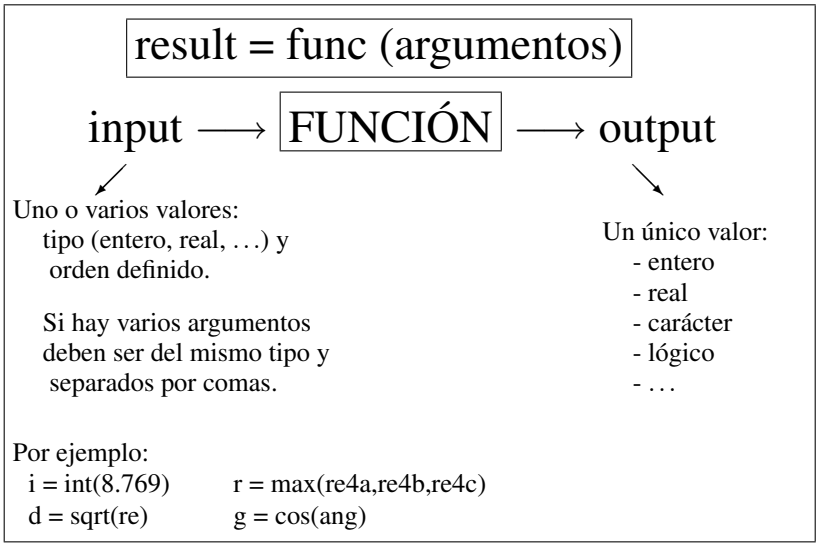


Figura 6: Funciones intrínsecas.

CUIDADO: Los argumentos de las funciones trigonométricas van siempre en radianes en vez de en grados. EDITE, COMPILE Y EJECUTE el programa prog3.f.

```
C  prog3.f               17 Oct 97
C  Autor: ....
C  Calcula el coseno de 45 grados y la raíz cuadrada de 36.
C = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
      ANG = 45.0
      RA  = 36.0

      PI = ACOS(-1.0)
      RAD = PI/180.0
      CS = COS (ANG*RAD)    ! Func. trigonométricas en radianes
      RAI= SQRT(RA)

      PRINT*, CS, RAI
      STOP
      END
```

Para saber cuanto vale la variable PI inserte debajo de la instrucción en donde se calcula el valor de PI la instrucción

PRINT*, PI

y vuelva a compilar y a ejecutar.

[C] ¿Para que se utiliza la variable RAD?.

Tabla 3: Funciones intrínsecas o predefinidas más habituales.

Nombre	Tipo argum.	Tipo función (output)	Observaciones
Funciones «matemáticas»:			
SQRT (gen)	Real o comp.	(1)	Raíz cuadrada
EXP (gen)	Real o comp.	(1)	Exponencial
LOG (gen)	Real o comp.	(1)	Logaritmo neperiano
LOG10 (gen)	Real	(1)	Logaritmo decimal
Funciones trigonométricas (trabajan en radianes):			
SIN (gen)	Real o comp.	(1)	Seno
COS (gen)	Real o comp.	(1)	Coseno
TAN (gen)	Real.	(1)	Tangente
ASIN (real)	Real.	(1)	Arcoseno
ACOS (real)	Real.	(1)	Arcocoseno
ATAN (real)	Real.	(1)	Arcotangente
Funciones de valor absoluto y signos:			
ABS (gen)	Int. real o comp.	(1)	Valor absoluto
SIGN (genA, genB)	Int. o real	(1)	Devuelve |genA| con el signo de genB.
Funciones de conversión:			
INT (gen)	Int. real o comp.	Int.	Devuelve el valor entero truncando los decimales.
REAL (gen)	Int. real o comp.	Real*4	Convierte un n° entero en real.
Funciones de redondeo y truncación:			
NINT (real)	Real	Int	Devuelve el entero por redondeo.
ANINT (real)	Real	(1)	Devuelve un número real sin decimales, redondeado.
Función modulo:			
MOD (genA, genB)	Int. o real	(1)	Devuelve el resto de dividir genA por genB.
Funciones de máximos y mínimos:			
MAX (genA, genB, ...)	Int. o real	(1)	Devuelve el mayor de los argumentos.
MIN (genA, genB, ...)	Int. o real	(1)	Devuelve el menor de los argumentos.

(1) Devuelve el mismo tipo de argumento que la entrada.

4. Bucles (DO)

Queremos sumar los diez primeros términos de la serie siguiente:

$$SUMA = \sum_{I=1}^{10} \frac{1}{I}$$

El programa **serie1.f** hace la suma de los diez términos, sin embargo, es bastante repetitivo.

Imagine cómo sería el programa si quisiéramos el sumatorio de los 100 o 1000 primeros términos de la serie. Para evitar la repetición del mismo tipo de instrucciones, el Fortran tiene la instrucción **do**. Con esta instrucción el programa anterior se simplifica como puede ver en el programa **bucle1.f**. La instrucción DO I = 2, 10 se entiende como «hacer desde I igual a 2 hasta I igual a 10 el conjunto de instrucciones que están entre el **do** y el **enddo**» (ver Fig. 7).

[C] En los programas serie1.f y bucle1.f ¿son necesarios los paréntesis?.

EDITE, COMPILE Y EJECUTE el programa bucle1.f. A continuación REEMPLACE la instrucción $R = R + (1.0/I)$ por $R = R + (1/I)$, ¿El resultado es el mismo?, ¿Por qué?.

```
C  serie1.f   26 Oct 09
C  Serie 1/I
C = = = = = = = = = = = =
      R = 1.0
      R = R + (1.0/2)
      R = R + (1.0/3)
      R = R + (1.0/4)
      R = R + (1.0/5)
      R = R + (1.0/6)
      R = R + (1.0/7)
      R = R + (1.0/8)
      R = R + (1.0/9)
      R = R + (1.0/10)

      PRINT*, ' SERIE: ', R
      STOP
      END
```

```
C   bucle1.f       26 Oct 09
C   Serie 1/I con un bucle
C = = = = = = = = = = = = = = = =
      R = 1.0

      DO I = 2, 10
      R = R + (1.0/I)
      ENDDO

      PRINT*, ' SERIE: ', R
      STOP
      END
```

Si ahora queremos calcular la suma de los 100 primeros términos tenemos que editar el programa bucle1.f, cambiar el 10 por un 100, volver a compilar y ejecutar. Para evitar el tener que editar y compilar el programa cada vez que queremos cambiar el valor límite de la serie tenemos la instrucción **read** (ver el programa buble2.f). Esta instrucción sirve para que el programa lea el contenido de una variable (en este caso la variable N) y lo guarde en memoria. Por tanto, al ejecutar el programa bucle2, el ordenador **se quedará esperando** a que tecleemos el valor que queremos que valga N. El programa lee este valor cuando después de teclearlo pulsamos la tecla intro y lo asigna a la variable N.

COPIE bucle1.f a **bucle2.f** utilizando la instrucción siguiente:

cp bucle1.f bucle2.f (↩)

EDITE bucle2.f y MODIFÍQUELO de acuerdo con la figura correspondiente a bucle2.f. COMPILE Y EJECUTE el programa bucle2.f. Observe la diferencia con respecto a bucle1.f.

```
C  bucle2.f              16 Oct 97     { 1}
C  Serie 1/I con un bucle y numero     { 2}
C  de terminos variables.              { 3}
C = = = = = = = = = = = = = = = = = =  { 4}
      PRINT*, ' Numero de terminos:'   { 5}
      READ*, N                         { 6}
      R = 1.0                          { 7}
                                       { 8}
      DO I = 2, N                      { 9}
      R = R + (1.0/I)                  {10}
      ENDDO                            {11}
                                       {12}
      PRINT*,' Terminos: ',N,          {13}
     +       ' Serie: ',R              {14}
      STOP                             {15}
      END                              {16}
```

ATENCIÓN: el carácter «+» o cualquier otro situado en la columna 6 indica que dicha línea es continuación de la línea anterior (ver Fig. 2), por lo tanto, en el ejemplo anterior, las líneas 13 y 14 equivalen a una sola instrucción:

PRINT*, 'Términos: ',N,' Serie: ',R

A continuación, ELIMINE la instrucción correspondiente al «print*, ' Numero ...» (línea 5) y vuelva a compilar y a ejecutar el programa bucle2.f. ¿Cual es la diferencia?.

Bucle básico:

do i = n1, n2 ... ... *enddo*	*do r = r1, r2* ... ... *enddo*	Esta instrucción se interpreta como: Haz el conjunto de instrucciones hasta el próximo **enddo** desde $i = n1 (r = r1)$ hasta $i = n2 (r = r2)$. $(i = n1, i = n1+1, i = n1+2, ..., i = n2)$

Observaciones:
- El índice del bucle es una variables entera («*i*») o real («*r*»).
- $n1, n2, r1$ y $r2$ pueden ser constantes, variables, o expresiones aritméticas (p.e. $4*n+2$).
- Si el límite superior ($n2$ o $r2$) es menor que el inferior ($n1$ o $r1$), el bucle se ignora.
- No se puede modificar el valor del indice «*i*» o «*r*» dentro del bucle.
- No se puede entrar en el bucle sin pasar por la instrucción **do**.

Figura 7: La instrucción **do**, bucle básico.

5. Decisiones (instrucción IF)

En determinadas ocasiones un programa debe determinar por sí mismo si realiza o no un conjunto de instrucciones concretas. Para ello el Fortran utiliza la instrucción **if** que podemos traducir como el «si» condicional. Si una determinada *condición* es cierta, entonces, ejecuta la instrucción (o conjunto de instrucciones). Si no (**else**) es cierta, entonces hace otra cosa.

Posibilidades de la instrucción **if**

Primera

if (expresión de valor lógico o condición) instrucción

Segunda

if (condición) **then**
...
...
endif

Tercera

if (condición) **then**
...
...
else
...
...
endif

Ejemplos:
- if (l <= 0) then
- if (u >B) then

- La «expresión de valor lógico o condición» va siempre entre paréntesis.

Figura 8: Tres posibilidades de la instrucción **if**.

En la Fig. 8 se muestran tres formas de utilizar esta instrucción. La expresión lógica o condición es una comparación de cuyo resultado determina, el programa, si realiza o no las instrucciones correspondientes. En el programa if1.f se utiliza la primera forma de la instrucción **if** para evitar un error al hacer la raíz cuadrada de un número negativo, estudie este programa (ver Tabla 4) y después compárelo con los programas if2.f e if3.f, que emplean la

segunda y tercera forma de la instrucción **if**, respectivamente.

EDITE, COMPILE y EJECUTE los tres programas. Cuando ejecute los programas, el ordenador se quedará esperando a que introduzca los valores de A y B para leerlos, ya que es esta la primera instrucción de los tres programas. Teclee ambos valores separados por uno o varios espacios en blancos (barra espaciadora) y pulse intro. No use las flechas de los cursores para avanzar y/o retroceder.

```
C if1.f       17 Oct 97
C Instruccion IF
C=====================
  READ*, A, B
  C = A - B
  IF(C <= 0.0) C = -1.0*C
  D = SQRT(C)
  PRINT*, A, B, C, D
  STOP
  END
```

```
C if2.f       17 Oct 97
C Instruccion IF
C ========================
   READ*, A, B
   C = A - B
   IF(C < 0.0) THEN
     C = -1.0*C
     PRINT*, ' N. complejo'
   ENDIF
   D = SQRT(C)
   PRINT*, A, B, C, D
   STOP
   END
```

```
C if3.f       17 Oct 97
C Instruccion IF
C ========================
   READ*, A, B
   C = A - B
   IF(C > 0.0) THEN
      D = SQRT(C)
      PRINT*, A, B, C, D
   ELSE
   D = SQRT(-1.0*C)
   PRINT*, ' N. complejo'
   PRINT*, A, B, C, D,'i'
   ENDIF
   STOP
   END
```

Tabla 4: Operadores (incluidos los puntos) utilizados en las comparaciones.

Operador		Significado
(Fortran 77)	(Fortran 90)	
.LT.	$<$	Menor que
.LE.	$<=$	Menor o igual que
.EQ.	$==$	Igual a
.NE.	$\backslash =$	No igual a
.GT.	$>$	Mayor que
.GE.	$>=$	Mayor o igual que

5.1. Introducción de datos por medio de un fichero.

EDITE un fichero llamado, por ejemplo, «ifdatos.dat» y escriba en él dos números separados por varios espacios en blanco; guarde el fichero y ejecute las instrucciones siguientes:

./if1.exe $<$ ifdatos.dat (↩)
./if2.exe $<$ ifdatos.dat (↩)
./if3.exe $<$ ifdatos.dat (↩)

Observe que cuando utilizamos el símbolo «$<$» los datos que nos piden los programas y que antes teníamos que introducir con el teclado ahora los lee del fichero «ifdatos.dat». También podemos escribir el resultado en un fichero; ejecute la instrucción:

./if3.exe $<$ ifdatos.dat $>$ ifsalida.sal (↩)

y edite el fichero «ifsalida.sal» para ver el resultado.

Tenga cuidado de no confundir los símbolos «$<$» y «$>$» pues si se equivoca puede borrar el fichero de datos. Los datos editados en el fichero de datos tiene que estar en la misma posición (separados por espacios y/o en diferentes líneas) que cuando los tecleamos en la pantalla. No tenemos que incluir comentarios en estos ficheros salvo que el programa Fortran lea dichos comentarios. En la sesión siguiente utilizaremos con frecuencia la introducción de datos por medio de un fichero.

6. GOTO (la instrucción que debemos evitar)

Observe el programa **num1.f**, en el introducimos nuevos elementos: la instrucción **goto** y las **etiquetas**. El **goto** (Fig. 9) significa «ve a ...», en este caso «ve a 10», donde el «10» situado delante de la instrucción **print** es una etiqueta.

Observe que las etiquetas tienen que ir siempre entre las columnas 1 y 5 (ver Fig. 2). Este programa en realidad es un modo alternativo de hacer un bucle pero sin la instrucción DO.

EDITE, COMPILE Y EJECUTE el programa num1.f.

```
C  num1.f                          17 Oct 97
C  Escribe numeros enteros del 1 al 100
C ====================================
      N = 1
  10  PRINT*, N
      N = N + 1
      IF(N <= 100) GOTO 10
      STOP
      END
```

goto etiqueta

Observaciones:
• La instrucción **goto** nunca puede llevar etiqueta.
• Después de una instrucción **goto** sólo puede haber una instrucción con etiqueta o un **end**.
• La instrucción correspondiente a la etiqueta puede estar delante o detrás de la instrucción **goto**.
• Conviene usar la instrucción **goto** lo menos posible, debido a que complica la estructura de los programas.

Figura 9: Instrucción básica del **goto**.

7. Ejercicio 1: Ecuación de segundo grado

El objetivo de este ejercicio consiste en programar la resolución de una ecuación de segundo grado:

$$ax^2 + bx + c = 0$$

Las soluciones (las raíces) de esta ecuación se obtienen mediante la fórmula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

Al resolver esta ecuación podemos encontrarnos algunos problemas. El primer problema se da cuando $a = 0$. En este caso no podemos emplear la ecuación anterior pues al dividir por cero en la Ec. 1 el programa nos dará un error. Evidentemente, en este caso, la ecuación a resolver sería

$$bx + c = 0\,; \qquad x = \frac{-c}{b}$$

Para simplificar hemos supuesto que a y b no valen ambos cero a la vez. El segundo problema es el cálculo de la raíz. Si su argumento ($b^2 - 4ac$) es negativo la raíz no tiene solución (dentro de los números reales) y el programa daría otro error. Conviene que nuestro programa compruebe estas posibilidades y evite los errores. Como vemos la programación se va complicando y conviene estructurar los programas antes de escribirlos en el ordenador. Para ello se utilizan los llamados diagramas de flujo. Un diagrama de flujo (no muy ortodoxo) para resolver la ecuación de segundo grado se muestra en la Fig. 10.

[P] Haga un programa que resuelva la ecuación de segundo grado. Ayúdese en el diagrama de flujo y las instrucciones dadas anteriormente. Utilice el programa para calcular las raíces de las siguientes ecuaciones particulares.
OPCIONAL: Incluir en el programa el caso de que las soluciones sean números complejos.

$2.2x^2 - 4.5x + 0.5 = 0\,;$ Solución: 1.9275 y 0.1179
$3.0x^2 + 3.0x + 0.75 = 0\,;$ Solución: -0.5 (doble)
$1.0x^2 + 2.0x + 1.5 = 0\,;$ Solución: $-1+0.707i$ y $-1-0.707i$

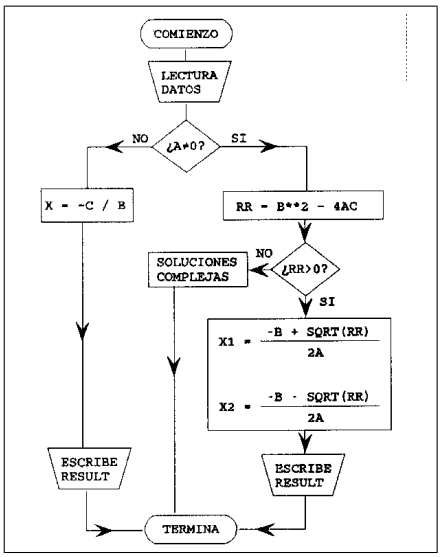


Figura 10: Diagrama de flujo para resolver una ecuación de segundo grado.

8. Problemas Adicionales (opcionales)

[P] El examen del espectro del átomo hidrógeno muestra la existencia de series de lineas espectrales, las frecuencias de las dos primeras series se calculan con las ecuaciones,

$$\text{Serie de Lyman:} \quad f = c \cdot R \cdot \left[\left(\frac{1}{1}\right)^2 - \left(\frac{1}{n}\right)^2 \right]; \quad n = 2,3,4,\ldots \quad \text{(ultravioleta)}$$

$$\text{Serie de Balmer:} \quad f = c \cdot R \cdot \left[\left(\frac{1}{2}\right)^2 - \left(\frac{1}{n}\right)^2 \right]; \quad n = 3,4,5,\ldots \quad \text{(visible)}$$

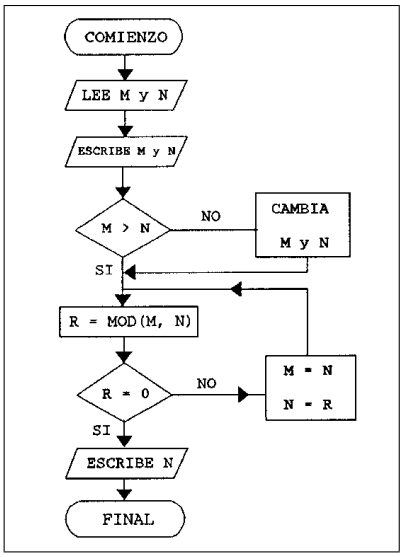
donde c es la velocidad de la luz ($2.998 \cdot 10^8$ m/s) y R es la constante de Rydberg ($1.0967758 \cdot 10^7$ m^{-1}). Escriba un programa Fortran para generar las primeras frecuencias de estas dos series.

[P] Escriba un programa Fortran para calcular el máximo común divisor (MCD) de dos números enteros (M y N) utilizando el algoritmo y diagrama de flujo de la figura y de acuerdo con las siguientes etapas:

a) Hacemos que el número M sea mayor que N, es decir si N es mayor que M intercambiamos los dos números.

b) Calculamos el resto (R) de dividir M por N. Si el resto (R) es cero entonces N es el MCD.

c) Si el resto (R) no es cero damos a M el valor de N y a N el valor de R y repetimos las etapas b) y c).



Referencias

[1] http://en.wikipedia.org/wiki/Fortran .

[2] Puedes obtener el gfortran en http://gcc.gnu.org/wiki/GFortran.

[3] Pedro Alberto Enriquez Palma y M^a Pilar Puyuelo García, *Introducción a la programación en Fortran 90 para Químicos. Prácticas*, Universidad de la Rioja, Servicio de Publicaciones, (2006). http://www.unirioja.es/servicios/sp/catalogo/online/fortran90.pdf.

[4] «Elementos de PROGRAMACIÓN FORTRAN para CIENTÍFICOS e INGENIEROS» de Manuel Fernández, Rosa Rodríguez, David Zorrilla y Jesús Sánchez. UCA (Editorial: Real Sociedad Española de Física, C.T.P.M 2006).

[5] Introducción al Fortran 90: http://www.cesga.es/telecursos/F90/.

[6] Curso breve de Fortran: http://www.uam.es/departamentos/ciencias/fisicateoricamateria/especifica/hojas/kike/FORTRAN/FORTRAN90.html.