	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	101/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Guía práctica de estudio 07: Fundamentos de Lenguaje FORTRAN



Elaborado por:


Ing. Jorge A. Solano Gálvez
Guadalupe Lizeth Parrales Romay

Revisado por:

M.C. Edgar E. García Cano

Autorizado por:

M.C. Alejandro Velázquez Mena

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	102/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Guía práctica de estudio 07: Fundamentos de lenguaje FORTRAN

Objetivo:

Elaborar programas en lenguaje FORTRAN utilizando las instrucciones de control de tipo *secuencia*.

Actividades:

- Declaración e impresión de variables de diferentes tipos de datos.
- Llamadas a funciones para la entrada (teclado) y salida (monitor) de datos.

Introducción

Una vez que un problema dado ha sido analizado (se identifican los datos de entrada y la salida deseada), que se ha diseñado un algoritmo que lo resuelva de manera eficiente (procesamiento de datos), y que se ha representado el algoritmo de manera gráfica o escrita (diagrama de flujo o pseudocódigo) se puede proceder a la etapa de codificación.

Dentro del ciclo de vida del software, la implementación de un algoritmo se encuentra dentro de la etapa de *codificación* del problema. Esta etapa va muy unida a la etapa de *pruebas*:

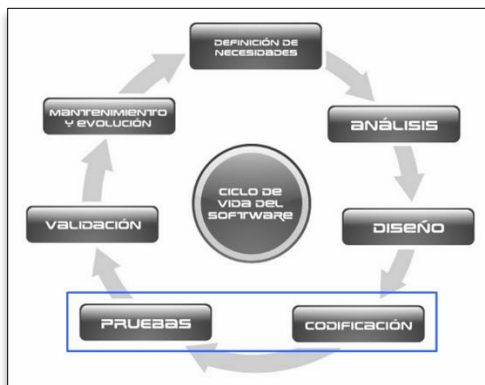



Figura 1: Ciclo de vida del software, resaltando las etapas de *codificación* y *pruebas*, las cuales se cubrirán en esta práctica.

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	103/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Lenguaje de programación FORTRAN

FORTRAN es uno de los lenguajes de programación más antiguos. Fue desarrollado por un equipo de programadores de IBM liderado por John Backus, y fue publicado por primera vez en 1957.

El nombre FORTRAN es un acrónimo para FORMula TRANslation debido a que fue diseñado para permitir la traducción de fórmulas matemáticas a código de forma fácil. FORTRAN fue el primer lenguaje de programación de alto nivel, y junto con el desarrollo de este lenguaje, también se desarrolló el primer compilador.


El objetivo del diseño de FORTRAN era crear un lenguaje de programación que fuera fácil de aprender, adecuado para una amplia variedad de aplicaciones, independiente de la plataforma y que permitiera manejar expresiones matemáticas complejas de forma similar a la notación algebraica regular.

FORTRAN es un lenguaje de propósito general basado en el paradigma imperativo. El paradigma de la programación imperativa consiste en determinar qué datos requiere el programa a realizar, asociar cada uno de estos datos a una dirección de memoria y realizar las operaciones necesarias sobre esos datos para llegar al resultado correcto.

FORTRAN es un lenguaje compilado, es decir, existe un programa (llamado compilador) que, a partir de un código en lenguaje FORTRAN, genera un código objeto (ejecutable).

Para crear un programa en FORTRAN se siguen tres etapas principales: edición, compilación y ejecución.

- Edición: Consiste en escribir el código fuente en lenguaje FORTRAN desde algún editor de textos.
- Compilación: A partir del código fuente (lenguaje FORTRAN) se genera el lenguaje máquina (se crea el código objeto o ejecutable).
- Ejecución: El archivo en lenguaje máquina se puede ejecutar en la arquitectura correspondiente (en la máquina donde se compiló el código fuente).

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	104/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Un programa en FORTRAN consiste en un programa principal y de ser necesario varios subprogramas (procedimientos, funciones o subrutinas).

Al momento de ejecutar un programa objeto (código binario), el compilador ejecutará únicamente las instrucciones que estén definidas dentro del programa principal. El programa principal puede contener sentencias y comentarios. Dentro de las sentencias se encuentran la declaración y/o asignación de variables, la realización de operaciones básicas y las llamadas a funciones.

Han surgido varias versiones de FORTRAN desde sus inicios, estas versiones son:

FORTRAN 66
FORTRAN 77
FORTRAN 90 (95)
FORTRAN 2003
FORTRAN 2008
FORTRAN 2010

Una de las versiones más utilizadas hoy en día sigue siendo FORTRAN 77.


Licencia GPL de GNU

El software presente en esta guía práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.

```

/*
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *

```

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	105/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

```
* You should have received a copy of the GNU General Public License
* along with this program. If not, see <http://www.gnu.org/licenses/>.
*
* Author: Jorge A. Solano
*
*/
```

Reglas de posición de columnas

FORTRAN no es un lenguaje libre de formato, existen un conjunto de reglas estrictas sobre el formato que debe tener el código fuente, las reglas más importantes son las de posición de columnas.

Col. 1: "c" o "*" para comentarios
Col. 1-5: Declaración de etiquetas (opcional)
Col. 6: Continuación de una línea previa (opcional)
Col. 7-72: Declaraciones
Col. 73-80: Secuencia de números (opcional y raramente utilizado)


Comentarios

Es una buena práctica en cualquier lenguaje de programación realizar comentarios para documentar el programa. En FORTRAN las líneas que comienzan con una 'c' o un asterisco (*) en la primera columna son considerados comentarios, algunos compiladores también aceptan el uso del signo '!' para comentarios.

Los comentarios pueden aparecer en cualquier parte del programa

Ejemplo

```
c Comentario por línea
* Otro comentario por línea
```

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	106/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código con comentarios

```

program nombre

* Este código compila y ejecuta
* pero no muestra salida alguna
! debido a que un comentario
! no es tomado en cuenta al momento
c de compilar el programa,
c solo sirve como documentación en el
! código fuente
    stop
end

```

NOTA. Un programa en FORTRAN debe comenzar con la palabra program seguida del nombre del programa y finalizar con las palabras stop y end, estas palabras comienzan en la columna 7.

Declaración de variables


Para declarar variables en FORTRAN se sigue la siguiente sintaxis:

```
tipoDeDato identificador
```

Por lo tanto, una variable debe declarar el tipo de dato que puede contener la variable y debe declarar el identificador (nombre o etiqueta) con el que se va a manejar el valor.

También es posible declarar varios identificadores de un mismo tipo de dato e inicializarlos en el mismo renglón, lo único que se tiene que hacer es separar cada identificador por comas.

```
tipoDeDato identificador1, identificador2
```

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	107/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Tipos de datos

Los tipos de datos básicos en FORTRAN son:

- Caracteres (character): codificación definida por la máquina (cadenas y caracteres simples).
- Enteros (integer): números sin punto decimal.
- Flotantes (real): números reales de precisión normal.
- Dobles (real*8 o double precisión): números reales de doble precisión.
- Complejos (complex): números complejos.
- Lógicos (logical): Permite representar los valores lógicos verdadero o falso.


Identificador

Un identificador es el nombre con el que se va a almacenar en memoria un tipo de dato. Los identificadores siguen las siguientes reglas:

- Debe iniciar con una letra [a-z].
- Puede contener letras [A-Z, a-z], números [0-9] y el carácter guión bajo (_).

NOTA: A pesar de que variables como 'areadeltriangulo' o 'perimetro_del_cuadrado' son declaraciones válidas como identificadores, es una buena práctica utilizar la notación de camello para nombrar las variables como convención.

En la notación de camello los nombres de cada palabra empiezan con mayúscula y el resto se escribe con minúsculas (a excepción de la primera palabra, la cual inicia también con minúscula). No se usan puntos ni guiones para separar las palabras. Además, las palabras de las constantes se escriben con mayúsculas y se separan con guion bajo.

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	108/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Para declarar constantes se utiliza la sentencia *parameter*, la sintaxis es la siguiente:

`parameter (NombreDeVariable = valor)`

Código declaración de variables

```

program tipos

c Este programa muestra la manera en la que se declaran y asignan variables
c de diferentes tipos: numéricas (enteras y reales), caracteres y lógicas.

c Variables enteras
integer numeroEntero
parameter (numeroEntero = 32768)


c Variables de tipo caracter
character caracterA
parameter (caracterA = 'a')

c Variables reales
real numeroFlotante
double precision numeroDoble
* Tambien se puede declarar una variable de doble precisión
* de la siguiente forma:
real*8 numeroDoble2

! Variables de tipo lógico
logical variableLogica
parameter (variableLogica = .TRUE.)

stop
end

```


	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	109/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

write es la función que se emplea para imprimir en la salida estándar (monitor) a través de lenguaje FORTRAN, la computadora le dará el formato más adecuado a los distintos tipos de variables, a excepción de las cadenas de caracteres:

```
write (*,*) 'El valor de la variable real es: ', varReal
```

read es una función que sirve para leer datos de la entrada estándar (teclado), para ello únicamente se especifica en qué variable se quiere almacenar el dato leído del teclado. El dato recibido se guardará en la localidad de memoria asignada a esa variable.

```
read (*,*) varEntera
```

Código almacenar e imprimir variables

```

program declaracion

c Este programa muestra la manera en la que
c se declaran y asignan variables
c de diferentes tipos: numéricas (enteras y reales)
c y caracteres, así como la manera en la que
c se leen e imprimen los diferentes tipos de datos.

* Variable entera
  integer numeroEntero


* Variable real
  real numeroReal
  double precision numeroDoble

* Variable de tipo caracter
  character caracterA

* Variable de tipo lógica
  logical variableLogica

! Asignar un valor a la variable caracterA

```

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	110/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

```

    parameter (caracterA = 'a')

! Asignar un valor a la variable caracterA
    parameter (variableLogica = .TRUE.)

! Asignar un valor del teclado a una variable

    write (*,*) 'Escriba un valor entero: '
    read (*,*) numeroEntero

    write (*,*) 'Escriba un valor real: '
    read (*,*) numeroReal

    write (*,*) 'Escriba un valor real de doble precisión: '
    read (*,*) numeroDoble

! Imprimir los valores obtenidos
    write (*,*) ''
    write (*,*) 'Imprimiendo la variable entera:'
    write (*,*) 'El valor de numeroEntero es: ', numeroEntero

    write (*,*) ''
    write (*,*) 'Imprimiendo el caracter:'
    write (*,*) 'El valor de caracterA es: ', caracterA


    write (*,*) ''
    write (*,*) 'Imprimiendo el valor lógico:'
    write (*,*) 'El valor de variableLogicaes: ', variableLogica

    write (*,*) ''
    write (*,*) 'Imprimiendo la variable real:'
    write (*,*) 'El valor de numeroReal es: ', numeroReal

    write (*,*) ''
    write (*,*) 'Imprimiendo la variable real de doble precisión:'
    write (*,*) 'El valor de numeroDoble es: ', numeroDoble

    write (*,*) ''
    stop
    end

```

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	111/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Tipo de dato complejo

El tipo de variable compleja, se define por un par de constantes reales separadas por coma y encerradas entre paréntesis, esta nomenclatura se aplica tanto a lectura desde la entrada estándar, como la lectura desde un archivo.

```

program complejo

c Programa que muestra el uso de los números complejos

complex a, b

write (*,*) 'Ingrese el valor de b'
read (*,*) b
a = (3.1, 4.7)
write (*,*) ''
write (*,*) 'Los valores de a y b son: '
write (*,*) 'a = ', a, ' b = ', b
write (*,*) ''
write (*,*) 'La suma de a + b es: '
write (*,*) 'a + b = ', a+b


stop
end program

```

Descriptores de formato

Los descriptores de formato sirven para dar un formato especial a un tipo de dato específico, ya sea para lectura o escritura. Los descriptores de formato de FORTRAN son:

I	Descriptor para datos de tipo entero
F	Descriptor para datos de tipo real
E	Descriptor de formato exponencial para tipos de dato reales
ES	Descriptor de formato científico para datos de tipo real
L	Descriptor para datos de tipo lógico
A	Descriptor para datos de tipo carácter
X	Descriptor especial de desplazamiento

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	112/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

T	horizontal (por columnas)
/	Descriptor especial de desplazamiento vertical (por líneas)

A continuación, se listan los símbolos usados con los descriptores de formato más comunes:

c	Número de columna.
d	Número de dígitos a la derecha del punto decimal para entrada/salida de datos reales.
m	Número mínimo de dígitos.
n	Número de espacios saltados
r	Factor de repetición: número de veces que se usa un descriptor o grupo de descriptores.
w	Anchura del campo: Número de caracteres de entrada/salida

La sintaxis para especificar un formato es la siguiente:

<pre>write (*, et) et format (descriptores)</pre>


En donde et es una etiqueta entera única, que puede tomar cualquier valor entre 1 y 99999.

Descriptor de formato para enteros (I)

La sintaxis para formato de salida es: [r]Iw[.m]

La sintaxis para formato de entrada es: [r]Iw

El valor se ajusta a la derecha del campo. Si el valor es demasiado grande para mostrarse con w caracteres, se muestran w asteriscos.

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	113/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

```

program DescriptorI

c Este programa muestra el uso del descriptor I

integer a, b, c

a = 134
b = -62
c = 0

c El siguiente formato imprime un entero de 6 dígitos
c si el número consta de menos de 6 dígitos, los dígitos
c faltantes se rellenan con espacios
write (*, 9) a
9 format (I6)

c Si se emplea el símbolo m, para definir el número
c mínimo de dígitos, en lugar de espacios los
c dígitos faltantes se rellenan con ceros
write (*, 10) a
10 format (I6.5)

c Si el número tiene más dígitos de los especificados
c en el descriptor, aparecerán asteriscos. El signo se
c considera como un dígito
write (*, 20) a
20 format (I2)


c b = -62, dado que el signo se considera como un
c dígito, el siguiente código mostrará dos asteriscos
write (*, 30) b
30 format (I2)

c Impresión de -62, con formato de 3 dígitos
write (*, 40) b
40 format (I3)

c Impresión de 0
write (*, 50) c
50 format (I2)

stop
end program

```

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	114/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Descriptor de formato para números reales (F)

La sintaxis para formato de entrada/salida es: [r]Fw.d

El valor se redondea para mostrar los dígitos de la parte decimal especificados con d. Si la parte decimal contiene menos dígitos que los especificados se rellenan los faltantes con ceros.

```

program DescriptorF

c Este programa muestra el uso del descriptor F

  real a, b, c, d

  a = 34.20
  b = -62.234
  c = 0.4739
  d = 0.0056

c Impresión en 9 caracteres con 4 decimales
  write (*, 9) a
  9 format (F9.4)

c Redondeo en 6 caracteres a 1 decimal
  write (*, 10) a
  10 format (F6.1)


  write (*, 11) c
  11 format (F6.1)

c Redondeo en 6 caracteres a 2 decimales
  write (*, 20) d
  20 format (F4.2)

c Impresión de un número negativo que requiere
c más caracteres de los especificados para su
c impresión
  write (*, 30) b
  30 format (F4.2)

c Impresión de un número negativo en 10 caracteres
c con tres decimales
  write (*, 31) b

```

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	115/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

31 format (F10.3)

```
stop
end program
```

Descriptor de formato para números reales en formato exponencial (E)

La sintaxis para formato de entrada/salida es: [r]Ew.d

Si se quiere leer/escribir un número real con d cifras decimales, la cantidad de caracteres necesarios para la impresión debe cumplir con la siguiente regla:

$$w \geq d+7$$

debido a que se requiere un carácter para representar el signo, al menos un carácter para la parte entera, un carácter para el punto decimal, otro para el carácter E, otro carácter para el signo de la parte exponencial y dos caracteres más para el exponente.

```
program DescriptorE

c Este programa muestra el uso del descriptor E


  real a, b
  a = 34.2047
  b = 0.0056

c Impresión de a en 9 caracteres con 2 decimales
  write (*, 9) a
  9 format (E9.2)

c Impresión de a en 6 caracteres con dos decimales
  write (*, 10) a
  10 format (E6.2)

c Impresión de b en 9 caracteres y 2 decimales
  write (*, 20) b
  20 format (E9.2)

c Impresión de b en 11 caracteres y 2 decimales
  write (*, 30) b
  30 format (E11.2)
```

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	116/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

```
stop
end program
```

Descriptor de formato para números reales en notación científica (ES)

Sintaxis para formato de entrada/salida: [r]ESw.d

Si se quiere leer/escribir un número real con d cifras decimales, la cantidad de caracteres necesarios para la impresión debe cumplir con la siguiente regla:

$$w \geq d+8$$

debido a que se requiere un carácter para representar el signo, al menos un carácter para la parte entera, un carácter para el punto decimal, un carácter para E y en algunos casos otro para S, otro carácter para el signo de la parte exponencial y dos caracteres más para el exponente.

```
program DescES


c Este programa muestra el uso del descriptor ES

real a, b
a = 34.2047
b = 0.0056

c Impresión de a en 11 caracteres con 3 decimales
write (*, 9) a
9 format (ES11.3)

c Impresión de a en 6 caracteres con dos decimales
write (*, 10) a
10 format (ES6.2)

c Impresión de b en 9 caracteres y 2 decimales
write (*, 20) b
```


	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	117/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

```
20 format (ES9.2)
```

```
c Impresión de b en 11 caracteres y 2 decimales
```

```
write (*, 30) b
```

```
30 format (ES11.2)
```

```
stop
```

```
end program
```

Descriptor de formato para datos de tipo lógico (L)

La sintaxis para formato de entrada/salida es: [r]Lw

La salida de un dato de tipo lógico será T para verdadero (TRUE) y F para falso (FALSE). No es muy común leer un dato de tipo lógico, sin embargo, se considera como el primer carácter no blanco que se encuentre dentro de la cantidad de caracteres definida por w.

```
program descL
```

```
logical a, b
```

```
a = .TRUE.
```

```
b = .FALSE.
```

```
c Imprime a en un espacio de 5 caracteres
```

```
write (*,10) a
```

```
10 format (L5)
```


```
c Imprime b en un espacio de 1 carácter
```

```
write (*,20) b
```

```
20 format (L1)
```

```
stop
```

```
end
```

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	118/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Descriptor de formato para datos de tipo caracter (A)

La sintaxis para formato de entrada/salida es: [r]A[w].

Si no aparece w, se lee/escribe el dato en un ancho igual a la longitud de la variable de tipo carácter.

```

program descA

character a
character (len=7) b

a = 'a'
b = 'hola'

c Especificando w
write (*,10) a
10 format(A5)

c Sin especificar w
write (*,20) b
20 format(A)


stop
end program

```

Operadores

Los operadores aritméticos que maneja lenguaje FORTRAN se describen en la siguiente tabla:

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>	<i>Resultado</i>
+	Suma	125.78 + 62.5	188.28
-	Resta	65.3 - 32.33	32.97
*	Multiplicación	8.27 * 7	57.75
/	División	15 / 4	3.75
**	Potenciación	4 ** 2	16

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	119/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Conversión de tipos de datos

Cuando en una expresión se presentan diferentes tipos de datos, es necesario hacer una conversión entre éstos. FORTRAN hace la conversión de datos de forma implícita en expresiones simples. Sin embargo, para expresiones complejas es necesario forzar las conversiones de forma explícita, las funciones disponibles para la conversión de tipos de datos son las siguientes:

<i>Función</i>	<i>Uso</i>
<i>int</i>	Convierte una variable numérica a entero
<i>real</i>	Convierte una entero a real
<i>dbble</i>	Convierte un entero a real de doble precisión
<i>ichar</i>	Convierte un caracter a entero
<i>char</i>	Convierte un entero al caracter equivalente en ASCII

Ejemplo:

```

program conversion

integer cinco, dos
double precision resImplicito, resExplicito
parameter (cinco = 5, dos = 2)


c La operación de división entre dos enteros genera
c un valor real, en este caso hay que moldear el
c resultado del lado derecho del igual para que
c que corresponda con el lado izquierdo y se pueda asignar

resImplicito = cinco/dos
resExplicito = dbble(cinco)/dbble(dos)

c El resultado con conversión implícita se trunca a 2.000
write (*,*) 'División con conversión implícita:', resImplicito
c El resultado con conversión explícita es el correcto 2.500
write (*,*) 'División con conversión explícita:', resExplicito

stop
end

```

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	120/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código operadores

```

program operadoresAritmeticos

c Este programa realiza las 5 operaciones
c aritméticas definidas en FORTRAN

integer ocho, cinco, tres, dos, uno
real res

parameter (ocho = 8, cinco = 5, tres = 3, dos = 2, uno = 1)

write (*,*) 'Operadores aritméticos'

res = real(cinco) / real(dos)
write (*,*) '5 / 2 = ', res

res = tres * ocho
write (*,*) '3 * 8 = ', res


res = dos + uno
write (*,*) '2 + 1 = ', res

res = ocho - cinco
write (*,*) '8 - 5 = ', res

res = dos ** cinco
write (*,*) '2 ** 5 = ', res

stop
end

```

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	121/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Expresiones lógicas

Las expresiones lógicas están constituidas por números, caracteres, constantes o variables que están relacionados entre sí por operadores lógicos. Una expresión lógica puede tomar únicamente los valores verdadero o falso.


Los operadores de relación permiten comparar elementos numéricos, alfanuméricos, constantes o variables.

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>	<i>Resultado</i>
.EQ.	Igual que	'h' .EQ. 'H'	Falso
.NE.	Diferente a	'a' .NE. 'b'	Verdadero
.LT.	Menor que	7 .LT. 15	Verdadero
.GT.	Mayor que	11 .GT. 22	Falso
.LE.	Menor o igual	15 .LE. 22	Verdadero
.GE.	Mayor o igual	20 .GE. 35	Falso

Los operadores lógicos permiten formular condiciones complejas a partir de condiciones simples.

<i>Operador</i>	<i>Operación</i>	<i>Uso</i>
.NOT.	No	.NOT. p
.AND.	Y	a > 0 .AND. a < 11
.OR.	O	opc == 1 .OR. salir != 0

NOTA: Lenguaje FORTRAN maneja los resultados booleanos (Verdadero o falso) como .TRUE. para verdadero y .FALSE. para falso

	Manual de prácticas del Laboratorio de Programación básica	Código:	MADO-18
		Versión:	01
		Página	122/184
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código expresiones lógicas

```

program operadoresLogicos

c Programa que muestra la forma en que se emplean los
c operadores lógicos en FORTRAN

integer num1, num2, num3, num4, res
character c1, c2

parameter (num1 = 7, num2 = 15, num3 = 3)
parameter (num4 = 3, c1 = 'h', c2 = 'H')

write (*,*) 'Expresiones de relación'

write (*,*) '{num1 es menor a num2?: ', num1 .LT. num2
write (*,*) '{num1 es mayor a num2?: ', num1 .GT. num2
write (*,*) '{c1 es igual a c2?: ', c1 .EQ. c2
write (*,*) '{c1 es diferente a c2?: ', num1 .NE. num2
write (*,*) '{num3 es menor a num4?: ', num3 .LE. num4
write (*,*) '{num1 es mayor a num3?: ', num1 .GE. num3

stop
end

```

Bibliografía

- Oracle (2010). Fortran 77 Lenguaje Reference. Consulta: Julio de 2015. Disponible en: <http://docs.oracle.com/cd/E19957-01/805-4939/>
- Stanford University (1995). Fortran 77 Tutorial. Consulta: Julio de 2015. Disponible en: http://web.stanford.edu/class/me200c/tutorial_77/
- Carlos Guadalupe (2013). Aseguramiento de la calidad del software (SQA). [Figura 1]. Consulta: junio de 2015. Disponible en: <https://www.mindmeister.com/es/273953719/aseguramiento-de-la-calidad-del-software-sqa>