



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	1/185
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

Elaborado por:	Revisado por:	Autorizado por:	Vigente desde:
Jorge A. Solano	Laura Sandoval Montaño	Alejandro Velázquez Mena	20 de enero de 2017

# **Manual de prácticas del laboratorio de Programación básica**



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	2/185
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

## Índice de prácticas

No	Nombre	Página
1	Búsquedas y utilerías en Internet	3
2	Solución de problemas	29
3	Algoritmos	38
4	Pseudocódigo	49
5	Diagramas de flujo	64
6	GNU/Linux	86
7	Fundamentos de Lenguaje FORTRAN	100
8	Estructuras de Selección	122
9	Estructuras de Repetición	133
10	Arreglos unidimensionales	147
11	Arreglos multidimensionales	158
12	Funciones	164
13	Lectura y escritura de datos	174



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	3/185
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 01: Búsquedas y utilerías en Internet



***Elaborado por:***

M.C. Edgar E. García Cano  
Ing. Jorge A. Solano Gálvez

***Revisado por:***

Ing. Laura Sandoval Montaño

***Autorizado por:***

M.C. Alejandro Velázquez Mena

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	4/185
		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 01: Búsquedas y utilerías en Internet

### **Objetivo:**

Realizar búsquedas especializadas, realizar visitas virtuales y conocer diferentes proveedores de servicios, todo a través de la web.

### **Actividades:**

- Realizar búsquedas especializadas a través de Internet.
- Conocer diferentes proveedores de servicios en Internet para realizar almacenamiento en la nube.
- Realizar visitas virtuales a través de diferentes sitios Web.

### **Introducción:**

Los motores de búsqueda (también conocidos como buscadores) son aplicaciones informáticas que rastrean la red de redes (Internet) catalogando, clasificando y organizando información, para poder mostrarla en el navegador.

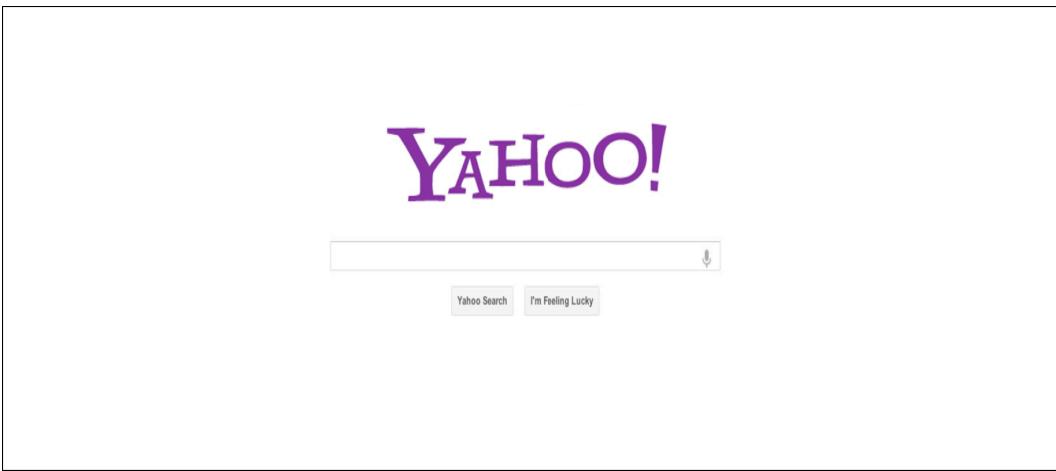
El rastreo de información se realiza a través de algoritmos propios de cada buscador, por ejemplo:

- Yahoo utiliza WebRank, a partir de una escala del 1 al 10, mide la popularidad de una página web.
- Bing utiliza un algoritmo que analiza diversos factores, como son el contenido de una página, el número y calidad de los sitios web que han enlazado la página, así como las palabras clave contenidas en el sitio.
- Google utilizar el llamado PageRank, que es un valor numérico que representa la popularidad que una página web tiene en Internet. PageRank es un concepto (marca registrada y patentada) de Google que introduce en su algoritmo de indexación.

### **Motores de Búsqueda en Internet**

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	5/185
		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			

El buscador **Bing**, propiedad de Microsoft fue puesto en línea en 2009. Desde 2011, el motor de búsqueda de Bing es utilizado por Yahoo! Search.



El motor de búsqueda DuckDuckGo (DDG) fue lanzado en 2008 bajo la política de no registrar información sobre las búsquedas del usuario.



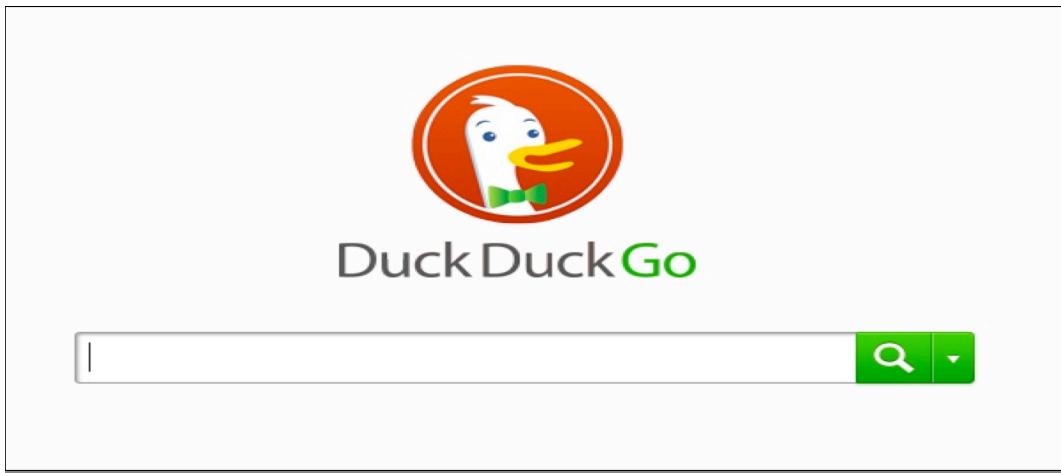
## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	6/185
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



El buscador de Google (en inglés Google Search) es un motor de búsqueda en la web propiedad de Google Inc. (Google). Es el motor de búsqueda más utilizado en la Web. Fue desarrollado por Larry Page y Sergey Brin en 1997.



A continuación, se muestran algunas de las características del motor de búsqueda de Google, las cuales sirven para mejorar las búsquedas, así como otras utilidades de las que carecen los otros motores de búsqueda y que pueden ser académicamente útiles. Se deja a consideración del alumno realizar las mismas búsquedas en los demás motores y decidir cuál es la opción que considere más útil.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	7/185
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

### Características

1. Para encontrar todas las imágenes de natación o de fútbol que no contengan la palabra tenis se utiliza la siguiente búsqueda:

The diagram shows a search bar containing the query "imagenes natacion or futbol -tenis". Handwritten annotations explain the operators: a yellow box labeled "minus" (-) indicates it should not contain the word "tenis"; a yellow box labeled "or" (or) indicates it should contain either "natacion" or "futbol".

**Nota:** no es necesario agregar acentos en la búsqueda.

2. Para encontrar todos los datos pertenecientes sólo a la **jornada del fútbol mexicano**:

The diagram shows a search bar containing the query "'jornada del futbol mexicano'". Handwritten annotations show double quotes at the beginning and end of the phrase, indicating an exact search.

Las comillas dobles ("<oración>") al inicio y al final de la búsqueda indican que sólo se deben buscar páginas que contengan exactamente dichas palabras. En este caso se agregó el conector *del* a la búsqueda para encontrar exactamente la frase.

3. Al momento de hacer búsquedas no es necesario incluir palabras como los artículos (el, la, los, las, un, etc.), pero en caso de ser necesario se puede hacer lo siguiente:



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	8/185
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

+la jornada

El símbolo de + sirve para que en la búsqueda se agregue la palabra y encuentre páginas que la incluyan.

### Comandos

Si se quiere saber el significado de una palabra, simplemente hay que agregar define:<palabra>.

define:computacion

site ayuda a buscar sólo en un sitio determinado.

site:cnnmexico.com ~olimpiadas 2012..2013

~ indica que encuentre cosas relacionadas con una palabra.

.. sirve para buscar en un intervalo de números, en este caso de años.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	9/185
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 realizar la búsqueda y obtener un tipo de documento en particular se usa **filetype:<tipo>**.

| intitle:"programación en c" intext:ingenieria filetype:pdf

**intitle:<palabra>** se encarga de encontrar páginas que tengan la palabra como título.

Para restringir los resultados donde se encuentre un término específico se usa **intext:<término>**.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	10/185
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

### Calculadora

Google permite realizar diversas operaciones dentro de la barra de búsqueda simplemente agregando la ecuación en dicho campo.

Google 5\*9 +(sqrt(10))^3

Web Imágenes Vídeos Libros Más Herramientas de búsqueda

Aproximadamente 84,900,000 resultados (0.39 segundos)

(5 \* 9) + (sqrt(10)^3) =

76.6227766017

Calculator interface showing the result 76.6227766017. The calculator has a numeric keypad and various mathematical function keys like Rad, Inv, sin, ln, etc.

Google sin(1) + cos(0)

Web Imágenes Vídeos Noticias Más Herramientas de búsqueda

Aproximadamente 37,300,000 resultados (0.33 segundos)

Sugerencia: [Buscar solo resultados en español](#). Puedes especificar el idioma de búsqueda en [Preferencias](#)

sin(1 radian) + cos(0 radians) =

1.84147098481

Calculator interface showing the result 1.84147098481. The calculator has a numeric keypad and various mathematical function keys like Rad, Inv, sin, cos, etc.

### Convertidor de unidades

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18	
		Versión:	01	
		Página	11/185	
		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				

El buscador de Google también se puede utilizar para obtener la equivalencia entre dos sistemas de unidades.



Google 90 grados centígrados a fahrenheit

Web Imágenes Vídeos Noticias Más Herramientas de búsqueda

Aproximadamente 112,000 resultados (0.46 segundos)

90 grados centígrados = **194 grados Fahrenheit**



Google 100 dólares a pesos

Web Imágenes Vídeos Noticias Más Herramientas de búsqueda

Aproximadamente 3,580,000 resultados (0.37 segundos)

100 dólares estadounidenses = **1 332.85351 pesos mexicanos**

Nota: el navegador interpreta la moneda nacional, si se requiere la conversión a otra moneda solo se especifica el tipo de peso (colombianos, argentinos, chilenos, etc.).

## Graficas en 2D

Es posible graficar funciones, para ello simplemente se debe insertar ésta en la barra de búsqueda. También se puede asignar el intervalo de la función que se desea graficar.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	12/185
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



### Graficas en 3D

Google también permite realizar gráficas en 3 dimensiones. A este tipo de gráficas es posible rotarlas para apreciarlas mejor.



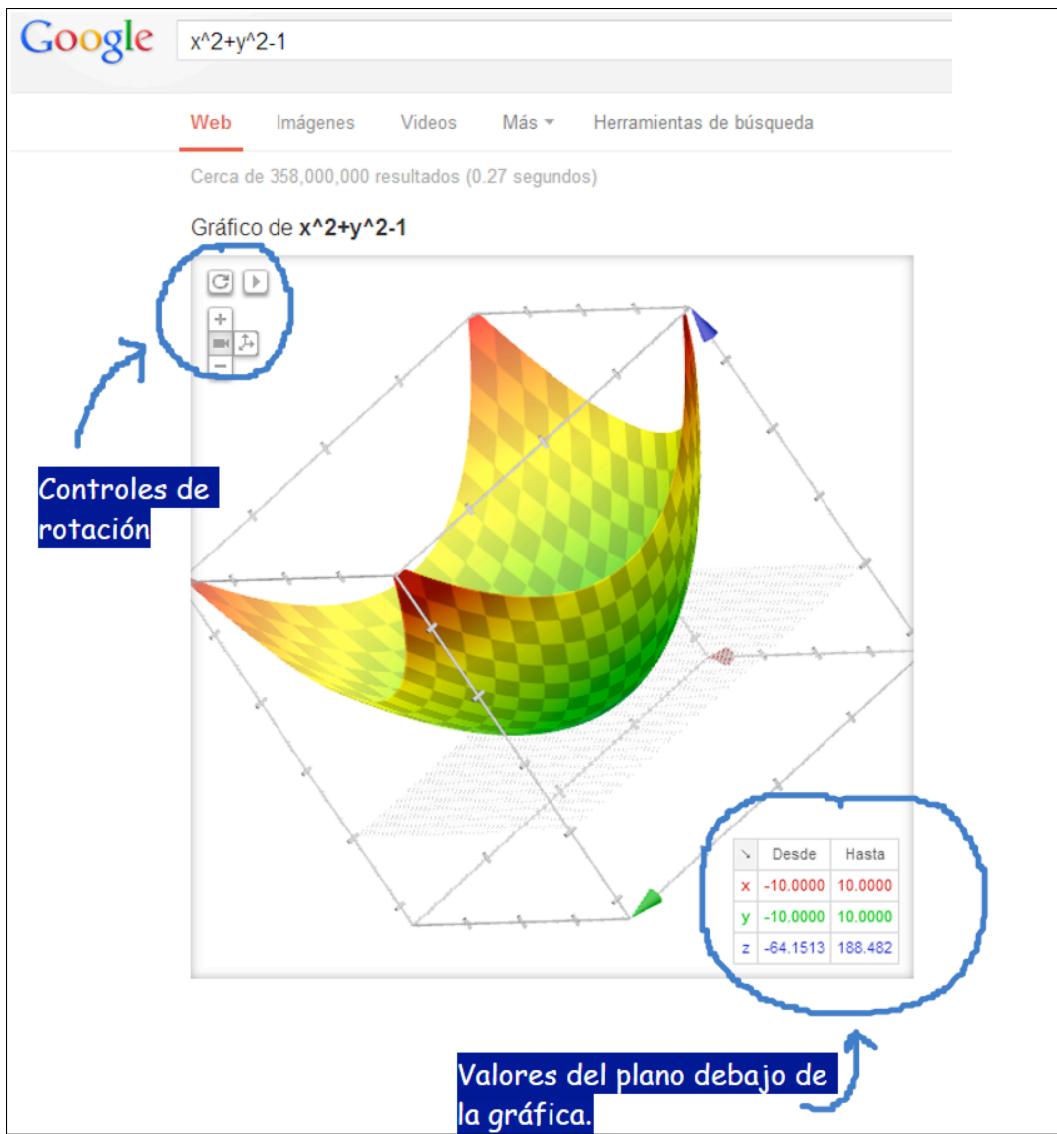
## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	13/185
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



Google académico

Si se realiza la siguiente búsqueda define:"google scholar", se obtiene:

"Google Académico es un buscador de Google especializado en artículos de revistas científicas, enfocado en el mundo académico, y soportado por una base de datos



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	14/185
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

disponible libremente en Internet que almacena un amplio conjunto de trabajos de investigación científica de distintas disciplinas y en distintos formatos de publicación."

<http://scholar.google.es/>



La siguiente búsqueda encuentra referencias del algoritmo de ordenamiento Quicksort creado por Hoare:

author:Hoare "quicksort"

Con el comando **author:<nombre>** se indica que se quiere buscar, artículos, libros y publicaciones de **un autor** en específico.

Dentro de la página se pueden observar varias características de la búsqueda realizada:



# Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	15/185
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

The screenshot shows a Google Scholar search results page. At the top left, there are filters: 'Artículos' (Articles), 'Mi biblioteca' (My library), 'Cualquier momento' (Anytime), 'Desde 2014', 'Desde 2013', 'Desde 2010', 'Intervalo específico...', 'Ordenar por relevancia' (Sort by relevance), 'Ordenar por fecha' (Sort by date), 'Buscar en la Web' (Search the Web), 'Buscar sólo páginas en español' (Search only Spanish pages), and two checked checkboxes: 'incluir patentes' (Include patents) and 'Rango de tiempo' (Time range). The search bar contains 'author:Hoare "quicksort"'. The results section has a heading 'Sugerencia: Buscar solo resultados en español' (Suggestion: Search only results in Spanish). It lists three main articles:

- Quicksort** (CAR Hoare - The Computer Journal, 1962 - Br Computer Soc)  
Abstract: A description is given of a new method of sorting in the random-access store of a computer. The method compares very favourably with other known methods in speed, in economy of storage, and in ease of programming. Certain refinements of the method, ...  
Citado por 866 Artículos relacionados Las 3 versiones Citar Guardar
- Proof of a recursive program: Quicksort** (M Foley, CAR Hoare - The Computer Journal, 1971 - Br Computer Soc)  
Abstract: This paper gives the proof of a useful and non-trivial program, Quicksort (Hoare, 1961). First, it is shown how to prove programs correct in a simple way, next a rigorous but informal proof of correctness is given, and finally some formal methods are introduced. ...  
Citado por 60 Artículos relacionados Las 2 versiones Citar Guardar Más
- Tipo de ordenamiento** (CAR Hoare - The origin of concurrent programming, 2002 - Springer)  
... Page 12. 242 CA R. HOARE operating in parallel on different elements of the same array ... Example: Quicksort Using this facility it is possible, if sufficient parallel hardware is available, to sort an array of size N in time proportional to N. ...  
Citado por 420 Artículos relacionados Las 4 versiones Citar Guardar Más

On the right side of the results, there is a link 'Sitio en el que está publicado.' (Published on the site) with a downward arrow pointing to '[PDF] de oxfordjournals.org' and another link '[PDF] de cmu.edu'.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	16/185
		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			

## Google imágenes

Permite realizar una búsqueda arrastrando una imagen almacenada en la computadora hacia el buscador de imágenes.

<http://www.google.com/imghp>





## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	17/185
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

The screenshot shows a Google search results page for the query "Koala.jpg". The search bar at the top has "Koala.jpg" and "mi pc" entered. Below the search bar, there are tabs for "Web", "Imágenes" (which is selected), "Videos", "Más", and "Herramientas de búsqueda". It displays approximately 1,970,000 results found in 0.50 seconds. The first result is a thumbnail of a koala, with its dimensions listed as 1024 x 768. Below the thumbnail, there's a link to "Mi PC Comunicaciones, Computadoras, Laptops, Impresoras y ..." and a snippet of text about MiPC.com.mx. Further down, there's a section titled "Imágenes similares - Notificar imágenes" showing several other koala images.

### Almacenamiento en la nube

El almacenamiento en la nube (o cloud storage, en inglés) es un modelo de servicio en el cual los datos de un sistema de cómputo se almacenan, se administran y se respaldan de forma remota, normalmente en servidores que están en la nube y que son administrados por el proveedor del servicio. Estos datos se ponen a disposición de los usuarios a través de una red, como lo es Internet.

Google Drive, SkyDrive, iCloud o Dropbox son algunos espacios de almacenamiento en la nube. Además, Google Drive (Google) y SkyDrive (Outlook) cuentan con herramientas que permiten crear documentos de texto, hojas de cálculo y presentaciones, donde el único requisito es tener una cuenta de correo de dichos proveedores.



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	18/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

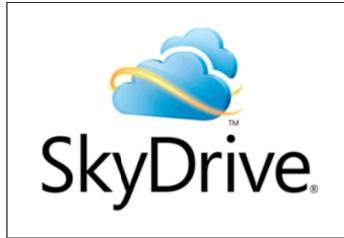
Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada



<http://www.youtube.com/watch?v=wKJ9KzGQq0w>



<http://www.youtube.com/watch?v=hoTBiIpz8DI>



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	19/185
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

Este tipo de herramientas hace posible editar un documento y compartirlo con uno o varios contactos, de tal manera que todos pueden trabajar grupalmente en un solo documento.



Por lo tanto, los documentos creados puedan ser vistos, editados, compartidos y descargados en cualquier sistema operativo, ya sea Windows, Mac OS o Linux, y en cualquier dispositivo con capacidad de procesamiento como teléfonos inteligentes, tabletas y computadoras.





## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	20/185
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

### Google Forms

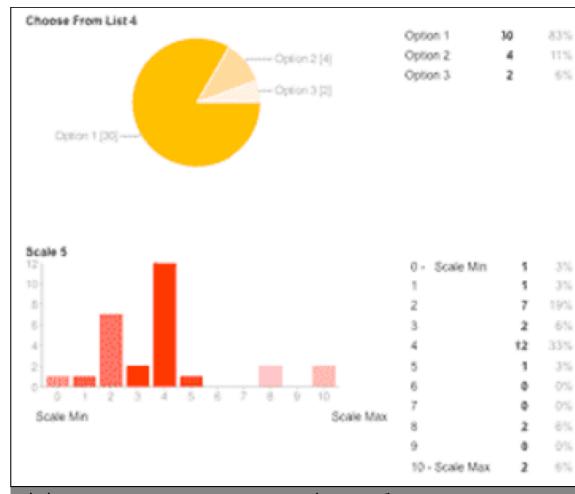
Google Drive cuenta con una aplicación para recolectar información usando formularios (Forms), una particularidad de la hoja de cálculo.

**New forms features**

What do you think about the new Forms features?

	This will change my life	Gee whiz, finally!	Pretty cool	Meh	I dislike change
Grid question type	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Bi-Di input support	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Improved results summary charts	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sign-in to view	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Pre-populate via parameter	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Se puede generar una serie de preguntas que pueden ser mandadas y contestadas por un grupo de personas. También proporciona un resumen con gráficas de los datos obtenidos del formulario.



<http://www.youtube.com/watch?v=IzgaUOW6GIs>

### OneNote

Por otro lado, a través de SkyDrive de Microsoft se puede utilizar la aplicación OneNote.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	21/185
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

El editor OneNote es muy amigable para realizar apuntes como si se ocupara una libreta de papel, pero con la diferencia de que todo se queda guardado en la nube.

The screenshot shows the Microsoft OneNote Web App interface. The top navigation bar includes File, Home, Insert, and View tabs. The ribbon below has sections for Clipboard, Basic Text, Styles, Tags, Spelling, and Office. On the left, a sidebar lists notebooks: Demo Notebook (selected), Cool-game Algorithms, Untitled Page, and Sailing around the world. The main content area displays a note titled "Sailing around the world" dated Wednesday, April 06, 2011, 1:40 PM. The note contains the text: "Ahoy there! This is my diary of my sailing adventures. Currently I'm sailing around the American continent. My trip started in Alaska and I'm planning stops along the way! Seattle, Oregon Beaches, San Francisco, just to name a few cities." Below the text is a world map with a dashed blue line representing a sailing route. A callout box with a sailboat icon points to the map, containing the handwritten note: "The boat shows my current location".

<http://www.youtube.com/watch?v=nxi9c6xBb0U>

## Dropbox

Dropbox es una herramienta que sirve para almacenar cualquier tipo de archivo digital en Internet.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	22/185
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 utilizarlo es necesario contar con una cuenta de correo para darse de alta en el sitio. Una vez realizado el registro se puede acceder al sitio, ya sea por medio de su interfaz web o descargando la aplicación que puede ser instalada en cualquier sistema operativo (teléfonos inteligentes, tabletas y computadoras).



Dropbox cuenta con aplicaciones de Microsoft Office Online para editar documentos. Los documentos también pueden ser compartidos con otros usuarios, ya sea compartiendo la carpeta que los contiene o por medio de un link.

<https://www.dropbox.com/>

### Visitas virtuales

Las visitas virtuales son una forma fácil e interactiva de recorrer un espacio, por medio de las “fotografías panorámicas esféricas”, que permiten observar el espacio fotografiado en 360° x 180°.

En una visita virtual el usuario percibe el espacio esférico con una vista totalmente verosímil y natural, desde cualquier parte del mundo a través de Internet. La idea



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	23/185
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

principal de un tour virtual es la de brindar al navegante la posibilidad de realizar una visita a un lugar físicamente distante con la sensación de estar allí.

A continuación, se mencionan algunos sitios que proveen visitas virtuales de diversos sitios.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	24/185
		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			

## Biblioteca Central de la UNAM

La Biblioteca Central de la UNAM cuenta con un recorrido virtual que permite recorrer el inmueble por dentro y por fuera.

[http://bc.unam.mx/cultural/inicio/vis\\_virt/main.html](http://bc.unam.mx/cultural/inicio/vis_virt/main.html)



	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	25/185
		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			

## Paseos virtuales del INAH

El Instituto Nacional de Antropología e Historia (INAH) brinda un mayor acercamiento a zonas arqueológicas, museos, exposiciones y bibliotecas a través de paseos virtuales.

<http://www.inah.gob.mx/index.php/catalogo-paseos-virtuales>

Destacamos: Zona Arqueológica de Toluquilla



El sitio arqueológico más grande de Querétaro, con cuatro canchas de juego de pelota; construcciones habitacionales, ceremoniales y administrativas, así como minas prehispánicas y modernas.

Street view de Google Inc



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	26/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

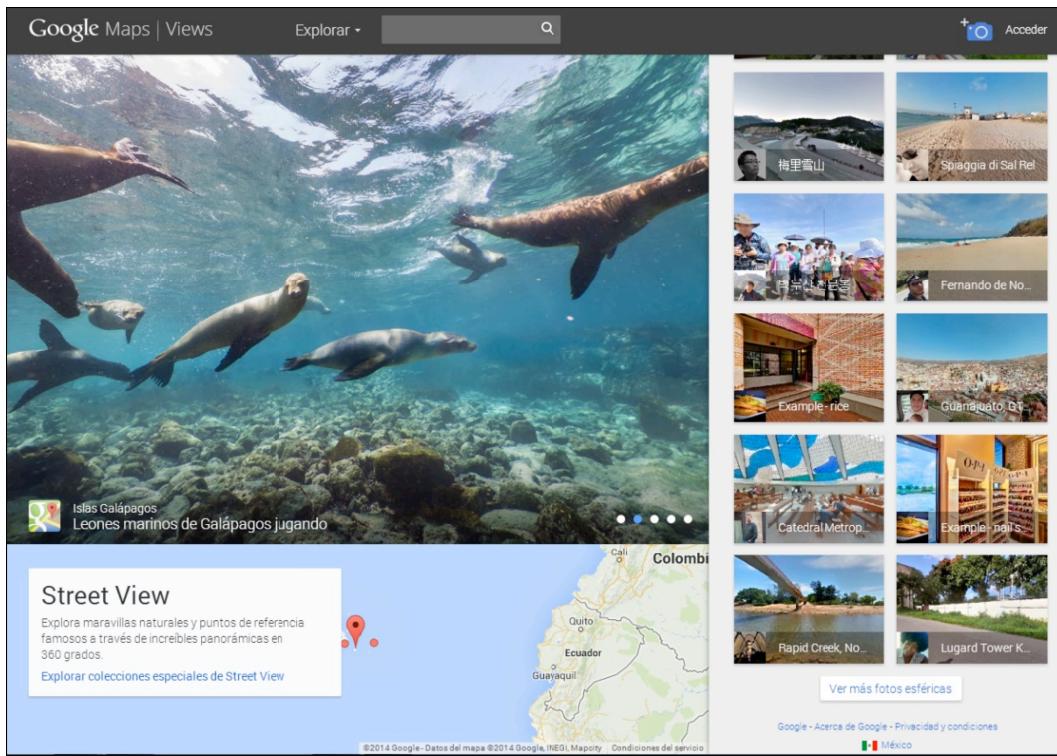
Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

Google provee el servicio de Street view el cual permite explorar maravillas naturales y monumentos famosos a través de fotos panorámicas de 360 grados.

<https://www.google.com/maps/views/home>



### Otros servicios de Google

Google ofrece varios servicios útiles para diversos fines: información, publicidad, consulta de libros, creación de usuarios, etc. Si se escribe google en la barra de búsqueda se obtienen todos los servicios que ofrece la compañía. A continuación, se listan algunos.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	27/185
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

- Google mapas: <https://maps.google.com/>
- Google traductor: <http://translate.google.com/>
- Google earth: <http://www.google.com/earth/>
- Google noticias: <http://news.google.com/>
- Google anuncios: <https://adwords.google.com/>
- Google libros: <http://books.google.com/>
- Google grupos: <https://groups.google.com/>

## Referencias

1. <https://www.bing.com>
2. <https://espanol.search.yahoo.com>
3. <https://duckduckgo.com>
4. <https://www.google.com.mx>
5. <http://scholar.google.es>
6. <http://www.google.com/imghp>
7. <http://www.youtube.com/watch?v=wKJ9KzGQq0w>



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	28/185
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

8. <http://www.youtube.com/watch?v=wKJ9KzGQq0w>
9. <http://www.youtube.com/watch?v=nxi9c6xBb0U>
10. <https://www.dropbox.com>
11. [http://bc.unam.mx/cultural/inicio/vis\\_virt/main.html](http://bc.unam.mx/cultural/inicio/vis_virt/main.html)
12. <http://www.inah.gob.mx/index.php/catalogo-paseos-virtuales>
13. <https://www.google.com/maps/views/home>
14. <https://maps.google.com>
15. <http://translate.google.com>
16. <http://www.google.com/earth>
17. <http://news.google.com>
18. <https://adwords.google.com>
19. <http://books.google.com>
20. <https://groups.google.com>



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	29/185
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 02: Solución de problemas



***Elaborado por:***

M.C. Edgar E. García Cano  
Ing. Jorge A. Solano Gálvez

***Revisado por:***

Ing. Laura Sandoval Montaño

***Autorizado por:***

M.C. Alejandro Velázquez Mena

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	30/185
		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 02: Solución de problemas

### **Objetivo:**

Identificar el conjunto de entrada (datos de entrada) y el conjunto de salida (datos de salida), a partir del análisis de la definición de un problema.

### **Actividades:**

- Identificar el conjunto de datos de entrada a partir de un problema dado.
- Identificar el conjunto de datos de salida a partir de un problema dado.

### **Introducción**

Un problema se puede definir como el conjunto de instancias al cual corresponde un conjunto de soluciones, junto con una relación que asocia para cada instancia del problema un subconjunto de soluciones (posiblemente vacío).

Para poder solucionar un problema es necesario averiguar qué es lo que requiere el usuario (análisis de requisitos). Esta etapa permite definir los requisitos de forma clara y concisa (especificación de requisitos).

En la creación de software, los desarrolladores se enfrentan al principio de un proyecto con un documento escrito por el cliente, en el cual expresa, en términos de la aplicación, qué se requiere del sistema. Este documento se conoce como declaración de requisitos.

### **Ingeniería de Software**

La Ingeniería de Software se define como el uso y establecimiento de principios de ingeniería sólidos, a fin de obtener un software que sea económicamente fiable y funcione eficientemente.

La Ingeniería de Software provee métodos que indican cómo generar software. Estos métodos abarcan una amplia gama de tareas:

- Planeación y estimación del proyecto.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	31/185
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

- Análisis de requerimientos del sistema y software.
- Diseño de la estructura de datos, la arquitectura del programa y el procedimiento algorítmico.
- Codificación.
- Pruebas y mantenimiento (validación y verificación).

## Ciclo de vida del software

La ISO (International Organization for Standardization) en su norma 12207 define al ciclo de vida de un software como:

Un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando desde la definición hasta la finalización de su uso.

Dentro del ciclo de vida del software, la solución del problema se encuentra dentro de la etapa de análisis de software:

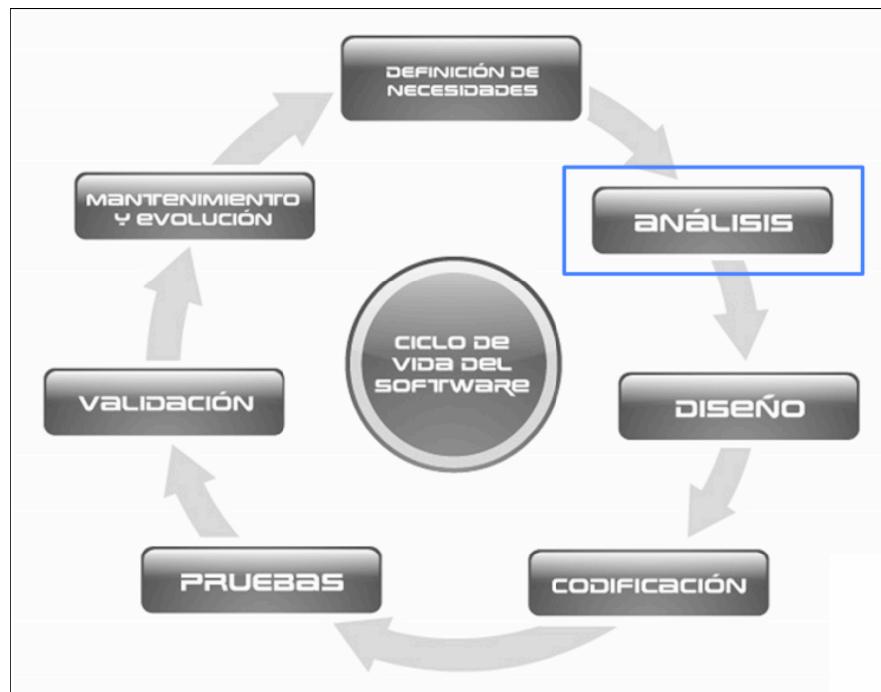


Figura 1: Ciclo de vida del software, resaltando la etapa de *análisis*, la cual corresponde a la solución de problemas.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	32/185
		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			

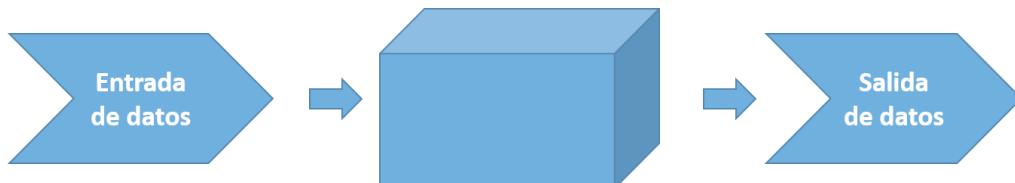
El análisis es el proceso para averiguar qué es lo que requiere el usuario del sistema de software (análisis de requisitos). Esta etapa permite definir las necesidades de forma clara y concisa (especificación de requisitos).

Por lo tanto, la etapa del análisis consiste en conocer qué es lo que está solicitando el usuario. Para ello es importante identificar dos grandes conjuntos dentro del sistema: el conjunto de entrada y el conjunto de salida.

El **conjunto de entrada** está compuesto por todos aquellos datos que pueden alimentar al sistema.

El **conjunto de salida** está compuesto por todos los datos que el sistema regresará como resultado del proceso. Estos datos se obtienen a partir de los datos de entrada.

La unión del conjunto de entrada y el conjunto de salida forman lo que se conoce como el dominio del problema, es decir, los valores que el problema puede manejar.



La etapa de análisis es crucial para la creación de un software de calidad, ya que si no se entiende qué es lo que se desea realizar, no se puede generar una solución. Sin embargo, es común caer en ambigüedades debido al mal entendimiento de los requerimientos iniciales.

A continuación, se presenta una imagen que representa los problemas más comunes en la administración de proyectos de software.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	33/185
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

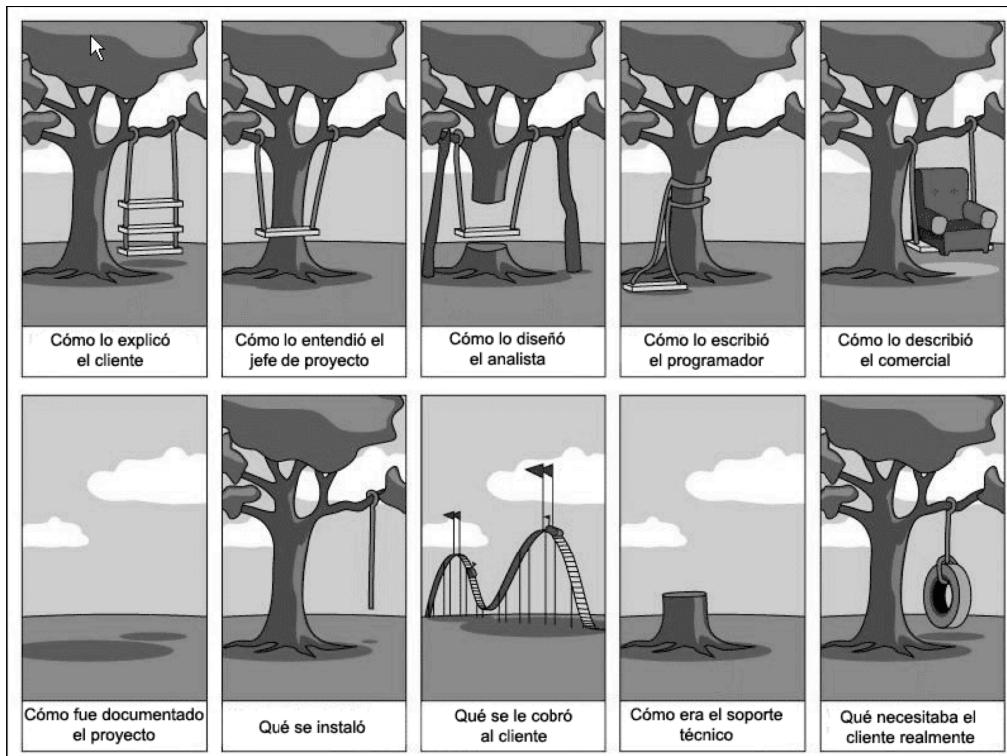


Figura 2: Fábula del columpio que describe las etapas de un proyecto contra el entendimiento en cada etapa.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	34/185
		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			

### Ejemplo 1

PROBLEMA: Determinar si un número dado es positivo o negativo.

RESTRICIONES: El número no puede ser cero.

DATOS DE ENTRADA: El conjunto de datos de entrada E está compuesto por el conjunto de los números reales, excepto el cero.

$$E \subset R^1, \text{ donde } num \in E \text{ de } (-\infty, \infty) - \{0\}$$

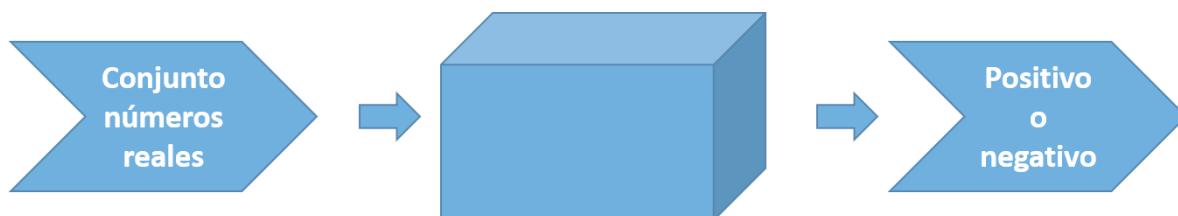
NOTA: R1 representa al conjunto de números reales de una dimensión.

DATOS DE SALIDA: El conjunto de salida S está compuesto por dos valores mutuamente excluyentes.

Un posible conjunto de salida son los valores enteros 0 o 1, donde 0 indica que el valor es positivo y 1 indica el valor es negativo.

$$res=0, \text{ si } num(0, \infty), res=1, \text{ si } num(-\infty, 0)$$

Otro posible conjunto de datos de salida son los valores booleanos *Verdadero* o *Falso*, donde *Verdadero* indica que el valor es positivo y *Falso* indica que el valor es negativo; o viceversa, *Verdadero* indica que el valor es negativo y *Falso* indica que el valor es positivo.



### Ejemplo 2

PROBLEMA: Obtener el mayor de dos números diferentes dados.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	35/185
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

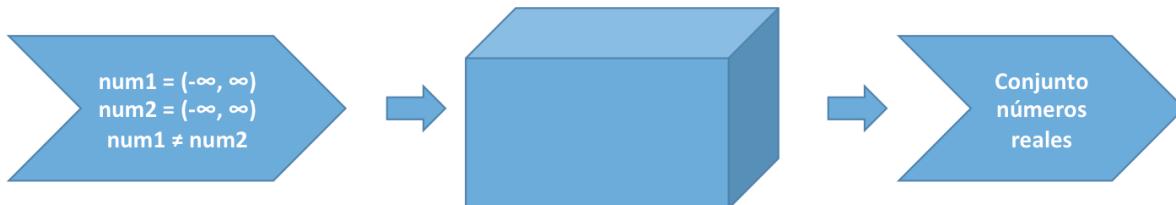
RESTRICCIONES: Los números de entrada deben ser diferentes.

DATOS DE ENTRADA: El conjunto de entrada E está dividido en dos subconjuntos E y E'. El primer número (num1) puede adquirir cualquier valor del conjunto de los números reales ( $E = (-\infty, \infty)$ ), sin embargo, el conjunto de entrada del segundo número (num2) es un subconjunto de E, es decir, E' está compuesto por el conjunto de los números reales excepto num1 ( $E' = (-\infty, \infty) - \{num1\}$ ).

$$E, E' \subset R^1, \text{ donde } num1 \in E \text{ de } (-\infty, \infty), num2 \in E' \text{ de } (-\infty, \infty) - \{num1\}$$

DATOS DE SALIDA: El conjunto de datos de salida S que puede tomar el resultado r está compuesto por el conjunto de los números reales.

$$S \subset R^1, \text{ donde } r \in S \text{ de } (-\infty, \infty)$$



### Ejemplo 3

PROBLEMA: Obtener el factorial de un número dado. El factorial de un número está dado por el producto de ese número por cada uno de los números anteriores hasta llegar a 1. El factorial de 0 (0!) es 1:

$$n! = n * (n-1)!$$

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	36/185
		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			

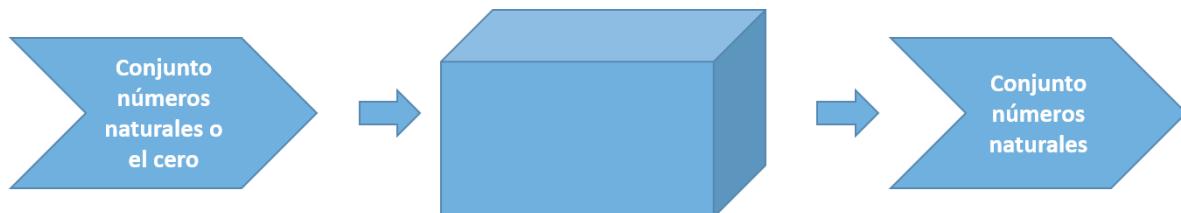
**RESTRICCIONES:** El número de entrada debe ser entero positivo o cero. No puede ser negativo.

**DATOS DE ENTRADA:** El conjunto de entrada E está dado por el conjunto de los números naturales o por el cero.

$$E \subset N^1, \text{ donde } num \in E \text{ de } \textcolor{red}{\text{de}} \textcolor{brown}{\text{de}} \cup \{0\}$$

**DATOS DE SALIDA:** El conjunto de salida S está conformado por el conjunto de los números naturales.

$$S \subset N^1; \text{ donde } res \in S \text{ de } \textcolor{red}{\text{de}} \textcolor{brown}{\text{de}}$$



## Referencias

- Raghu Singh (1995). International Standard ISO/IEC 12207 Software Life Cycle Processes. Agosto 23 de 1996, de ISO/IEC. Consulta: junio de 2015. Disponible en: <http://www.abelia.com/docs/12207cpt.pdf>
- 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>



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	37/185
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

- Andrea S. (2014). Ingeniería de Software. [Figura 2]. Consulta: Junio de 2015. Disponible en: <http://ing-software-verano2014.blogspot.mx>



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	38/185
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 03: Algoritmos



***Elaborado por:***

M.C. Edgar E. García Cano  
Ing. Jorge A. Solano Gálvez

***Revisado por:***

Ing. Laura Sandoval Montaño

***Autorizado por:***

M.C. Alejandro Velázquez Mena

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	39/185
		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 03: Algoritmos

### **Objetivo:**

A partir del análisis del problema (conjuntos de entrada y salida), elaborar algoritmos que permitan resolver el problema planteado.

### **Actividades:**

- Analizar problemas.
- Crear algoritmos para resolver problemas.

### **Introducción**

Una vez realizado el análisis, es decir, ya que se entendió qué es lo que está solicitando el usuario y ya identificado el conjunto de entrada y el conjunto de salida, se puede proceder al diseño de la solución, esto es, a la generación del algoritmo.

Un problema matemático es computable si éste puede ser resuelto, en principio, por un dispositivo computacional.

La teoría de la computabilidad es la parte de la computación que estudia los problemas de decisión que pueden ser resueltos con un algoritmo.

Un algoritmo es un conjunto de reglas, expresadas en un lenguaje específico, para realizar alguna tarea en general, es decir, un conjunto de pasos, procedimientos o acciones que permiten alcanzar un resultado o resolver un problema. Estas reglas o pasos pueden ser aplicados un número ilimitado de veces sobre una situación particular.

Un algoritmo es la parte más importante y durable de las ciencias de la computación debido a que éste puede ser creado de manera independiente tanto del lenguaje como de las características físicas del equipo que lo va a ejecutar.

Las principales características con las que debe cumplir un algoritmo son:



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	40/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

Facultad de Ingeniería

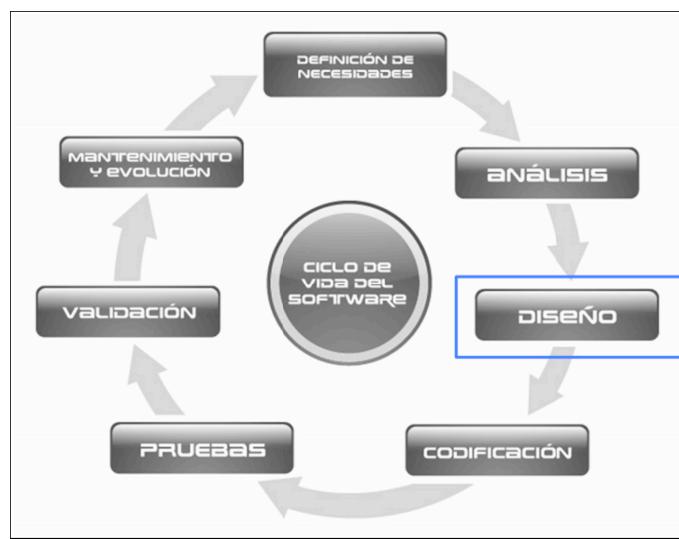
Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

- Un algoritmo debe ser **preciso**, es decir, llegar a la solución en el menor tiempo posible y sin ambigüedades.
- También debe ser **determinista**, es decir, a partir de un conjunto de datos idénticos de entrada, debe arrojar siempre los mismos resultados a la salida.
- El que un proceso sea computable implica que, en algún momento, el proceso va a llegar a su fin. Un algoritmo debe ser **finito**, por tanto, en algún momento debe terminar, lo que al mismo tiempo implica que el dominio del problema debe estar acotado.

Por tanto, un buen algoritmo debe ser correcto (cumplir con el objetivo) y eficiente (realizarlo en el menor tiempo posible), además de ser entendible para cualquier persona.

Dentro del ciclo de vida del software, la creación de un algoritmo se encuentra en la etapa de diseño de la solución del problema:



**Figura 1:** Ciclo de vida del software, resaltando la etapa de *diseño*, la cual corresponde al diseño de algoritmos.

Un algoritmo consta de 3 módulos básicos: módulo de entrada, módulo de procesamiento y módulo de salida.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	41/185
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



El **módulo de entrada** representa los datos que requieren para resolver el problema. Estos datos se pueden solicitar al usuario, leer de un archivo, consultar de una base de datos, etc.

El **módulo de procesamiento** de datos representa las operaciones necesarias para obtener un resultado a partir de los datos de entrada.

El **módulo de salida** permite mostrar los resultados obtenidos a partir del módulo de procesamiento de datos. Los resultados pueden mostrarse en diversos sitios: en la pantalla, en un archivo, en una base de datos, etc.

Para saber si un algoritmo es adecuado dos preguntas se vuelven fundamentales:

1. ¿Es correcto?
2. ¿Es eficiente?

Para poder afirmar que un algoritmo es correcto (cumple con el objetivo del problema) es necesario aplicarlo con varios elementos del conjunto de entrada (datos de entrada), realizando una prueba de escritorio para verificar que, para todos los datos del conjunto de entrada, se obtiene la salida esperada (datos del conjunto de salida).

Se puede considerar que un algoritmo es eficiente cuando el tiempo de ejecución es el menor posible para cualquier dato del conjunto de entrada.

Por lo anterior, el siguiente se podría definir como un algoritmo para crear un algoritmo:

1. Analizar el problema a resolver (obtener los conjuntos de entrada y salida de datos).
2. Idear un algoritmo que solvete el problema planteado.
3. Verificar que el algoritmo planteado sea correcto, es decir, que resuelva el problema.
4. Analizar la eficiencia del algoritmo.
5. Si el algoritmo planteado no es correcto o si el



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	42/185
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

algoritmo

planteado no es eficiente, se regresa al punto 2.

6. Si el algoritmo planteado es correcto y es eficiente, se termina el proceso.

Fuente: [udacity.com, Intro to Algorithms](https://www.udacity.com/course/viewer# !/c-cs215/l-48747095/m-48691609),

<https://www.udacity.com/course/viewer# !/c-cs215/l-48747095/m-48691609>



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	43/185
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

### Ejemplo 1

PROBLEMA: Determinar si un número dado es positivo o negativo.

RESTRICIONES: El número no puede ser cero.

DATOS DE ENTRADA: Número real.

DATOS DE SALIDA: La validación de si el número es positivo

DOMINIO: Todos los números reales.

### SOLUCIÓN:

1. Solicitar un número real.
2. Si el número ingresado es cero, se regresa al punto 1.
3. Si el número ingresado es diferente de cero, se validan las siguientes condiciones:
  - 3.1 Si el número ingresado es mayor a 0 se puede afirmar que el número es positivo.
  - 3.2 Si el número ingresado es menor a 0 se puede afirmar que el número es negativo.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	44/185
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

Prueba de escritorio

El diseño de la solución de un problema implica la creación del algoritmo y la validación del mismo. La validación se suele realizar mediante una *prueba de escritorio*.

Una prueba de escritorio es una matriz formada por los valores que van adquiriendo cada una de las variables del programa en cada iteración. Una iteración es el número de veces que se ejecuta un código y permite ver los valores que van adquiriendo las variables en cada repetición.

Para el ejemplo en cuestión la prueba de escritorio quedaría de la siguiente manera:

Iteración	X	Salida
1	5	El número 5 es positivo

Iteración	X	Salida
1	-29	El número 29 es negativo

Iteración n	X	Salida
1	0	-
2	0	-
3	0	-
4	100	El número 100 es positivo



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	45/185
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

### Ejemplo 2

PROBLEMA: Obtener el mayor de dos números dados.

RESTRICIONES: Los números de entrada deben ser diferentes.

DATOS DE ENTRADA: Número real.

DATOS DE SALIDA: La impresión del número más grande.

DOMINIO: Todos los números reales.

### SOLUCIÓN:

1. Solicitar un primer número real.
2. Solicitar un segundo número real.
3. Si el segundo número real es igual al primer número real, se regresa al punto 2.
4. Si el segundo número real es diferente al primer número real, se validan las siguientes condiciones:
  - 4.1 Si se cumple con la condición de que el primer número es mayor al segundo número, entonces se puede afirmar que el primer número es el mayor de los números.
  - 4.2 Si se cumple con la condición de que el segundo número es mayor al primer número, entonces se puede afirmar que el segundo número es el mayor de los números.

Prueba de escritorio:

Iteración	X	Y	Salida
1	5	6	El número 6 es el mayor

Iteración	X	Y	Salida
1	-99	-222.2	El número -99 es el mayor



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	46/185
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

Iteración	X	Y	Salida
1	15	15	-
2	15	15	-
3	15	15	-
4	15	10	El número 15 es el mayor

### Ejemplo 3

PROBLEMA: Obtener el factorial de un número dado. El factorial de un número está dado por el producto de ese número por cada uno de los números anteriores hasta llegar a 1. El factorial de 0 ( $0!$ ) es 1.

RESTRICIONES: El número de entrada debe ser entero y no puede ser negativo.

DATOS DE ENTRADA: Número entero.

DATOS DE SALIDA: La impresión del factorial del número.

DOMINIO: Todos los números naturales positivos.

### SOLUCIÓN:

1. Solicitar un número entero.
2. Si el número entero es menor a cero regresar al punto 1.
3. Si el número entero es mayor a cero se crea una variable entera *contador* que inicie en 2 y una variable entera *factorial* que inicie en uno.
4. Si la variable *contador* es menor o igual al número entero se realiza lo siguiente:
  - 4.1 Se multiplica el valor de la variable *contador* con el valor de la variable *factorial*. El resultado se almacena en la variable *factorial*.
  - 4.2 Se incrementa en uno el valor de la variable *contador*.
5. Si la variable *contador* no es menor o igual al número entero se muestra el resultado almacenado en la variable *factorial*.



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	47/185
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



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	48/185
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

Prueba de escritorio:

Iteración	X	fact	i	Salida
1	0	1	2	El factorial de 0 es: 1

Iteración	X	fact	i	Salida
1	-2	1	2	-
2	-67	1	2	-
3	5	1	2	-
4	5	2	3	-
5	5	6	4	-
6	5	24	5	-
7	5	120	6	El factorial de 5 es: 120

Iteración	X	fact	i	Salida
1	7	1	2	-
2	7	2	3	-
3	7	6	4	-
4	7	24	5	-
5	7	120	6	-
6	7	720	7	-
7	7	5040	8	El factorial de 7 es: 5040

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	49/185
		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			

## Referencias

- 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>
- Michael Littman. (2012). Intro to Algorithms: Social Network Analysis. Consulta: Junio de 2015, de Udacity. Disponible en: <https://www.udacity.com/course/viewer#!/c-cs215/l-48747095/m-48691609>



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	50/185
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 04: Pseudocódigo



***Elaborado por:***

M.C. Edgar E. García Cano  
Ing. Jorge A. Solano Gálvez

***Revisado por:***

Ing. Laura Sandoval Montaño

***Autorizado por:***

M.C. Alejandro Velázquez Mena

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	51/185
		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 04: Pseudocódigo

### **Objetivo:**

Elaborar pseudocódigos que representen soluciones algorítmicas empleando la sintaxis y semántica adecuadas.

### **Actividades:**

- Analizar un problema.
- Crear un algoritmo para resolver el problema.
- Representar el algoritmo en pseudocódigo.

### **Introducción**

Una vez que un problema dado ha sido analizado (se obtiene el conjunto de datos de entrada y el conjunto de datos de salida esperado) y se ha diseñado un algoritmo que lo resuelva de manera eficiente (procesamiento de datos), se debe proceder a la etapa de codificación del algoritmo.

Para que la solución de un problema (algoritmo) pueda ser codificada, se debe generar una representación del mismo. Una representación algorítmica elemental es el pseudocódigo.

Un pseudocódigo es la representación escrita de un algoritmo, es decir, muestra en forma de texto los pasos a seguir para solucionar un problema. El pseudocódigo posee una sintaxis propia para poder realizar la representación del algoritmo (solución de un problema).

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	52/185
		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			

## Sintaxis de pseudocódigo

El lenguaje pseudocódigo tiene diversas reglas semánticas y sintácticas. A continuación, se describen las más importantes:

1. Alcance del programa: Todo pseudocódigo está limitado por las etiquetas de INICIO y FIN. Dentro de estas etiquetas se deben escribir todas las instrucciones del programa.
1. Palabras reservadas con mayúsculas: Todas las palabras propias del pseudocódigo deben de ser escritas en mayúsculas.
1. Sangría o tabulación: El pseudocódigo debe tener diversas alineaciones para que el código sea más fácil de entender y depurar.
1. Lectura / escritura: Para indicar lectura de datos se utiliza la etiqueta LEER. Para indicar escritura de datos se utiliza la etiqueta ESCRIBIR. La lectura de datos se realiza, por defecto, desde el teclado, que es la entrada estándar del sistema. La escritura de datos se realiza, por defecto, en la pantalla, que es la salida estándar del sistema.

### Ejemplo

```
ESCRIBIR "Ingresar la altura del polígono"
LEER altura
```

1. Declaración de variables: la declaración de variables la definen un identificador (nombre), seguido de dos puntos, seguido del tipo de dato, es decir:

```
<nombreVariable>:<tipoDeDato>
```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	53/185
		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			

Los tipos de datos que se pueden utilizar son:

ENTERO -> valor entero positivo y/o negativo  
 REAL -> valor con punto flotante y signo  
 BOOLEANO -> valor de dos estados: verdadero o falso  
 CARACTER -> valor tipo carácter  
 CADENA -> cadena de caractéres

### Ejemplo

```

    contador: ENTERO
    producto: REAL
    continuar: BOOLEANO
  
```

Es posible declarar más de una variable de un mismo tipo de dato utilizando arreglos, indicando la cantidad de variables que se requieren, su sintaxis es la siguiente:

```
<nombreVariable>[cantidad]:<tipoDeDato>
```

### Ejemplo

```

    contador[5]: ENTERO      // 5 variables de tipo entero
    division[3]: REAL        // 3 variables de tipo real
    bandera[6]: BOOLEANO    // 6 variables de tipo booleano
  
```

Existe un tipo de dato compuesto, es decir, que puede contener uno o más tipos de datos simples diferentes. Este tipo de dato se conoce como registro o estructura y su sintaxis es la siguiente

```

<nombreRegistro>:REG
  <nombreVariable_1>:<tipoDeDato>
  ...
  <nombreVariable_N>:<tipoDeDato>
FIN REG
  
```

Para crear una variable tipo registro se debe indicar el nombre del registro y el nombre de la variable. Para acceder a los datos del registro se hace uso del operador "..".

### Ejemplo

```
domicilio:REG
```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	54/185
		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			

calle: STR  
 número: ENTERO  
 ciudad: STR  
 FIN REG

```
usuario:REG domicilio // variable llamada usuario de tipo registro
usuario.calle := "Av. Imán"
usuario.numero := 3000
usuario.ciudad := "México"
```

Es posible crear variables constantes con la palabra reservada CONST, la cual indica que un identificador no cambia su valor durante todo el pseudocódigo. Las constantes (por convención) se escriben con mayúsculas y se deben inicializar al momento de declararse.

### Ejemplo

1. NUM\_MAX := 1000: REAL, CONST  
 Operadores aritméticos: Se tiene la posibilidad de utilizar operadores aritméticos y lógicos:

Operadores aritméticos: suma (+), resta (-), multiplicación (\*), división real (/), división entera (div), módulo (mod), exponentiación (^), asignación (:=).

Operadores lógicos: igualdad (=), y-lógica o AND (&), o-lógica u OR (|), negación o NOT (!), relaciones de orden (<, >, <=, >=) y diferente (<>).



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	55/185
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

La tabla de verdad de los operadores lógicos AND, OR y NOT se describe a continuación:

A	B	A & B	A   B	!A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

NOTA: A y B son dos condiciones, el valor 0 indica falso y el valor 1 indica verdadero.

- Notación de camello. Para nombrar variables y nombres de funciones se debe hacer uso de la notación de camello.

En la notación de camello (llamada así porque parecen las jorobas de un camello) los nombres de cada palabra empiezan con mayúscula y el resto se escribe con minúsculas. Existen dos tipos de notaciones de camello: lower camel case que en la cual la primera letra de la variable inicia con minúscula y upper camel case en la cual todas las palabras inician con mayúscula. No se usan puntos ni guiones para separar las palabras (a excepción de las constantes que utilizan guiones bajos). Además, para saber el tipo de variable se recomienda utilizar un prefijo.

### Ejemplo

```
// variables
realAreaDelTriangulo: REAL // lower camel case
EnterRadioCirculo: REAL // upper camel case

// funciones
calcularArea()
obtenerPerimetro()
```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	56/185
		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			

## Estructuras de control de flujo

Las estructuras de control de flujo permiten la ejecución condicional y la repetición de un conjunto de instrucciones.

Existen 3 estructuras de control: secuencial, condicional y repetitivas o iterativas.

### Estructura de control secuencial

Las estructuras de control secuenciales son las sentencias o declaraciones que se realizan una a continuación de otra en el orden en el que están escritas.

#### *Ejemplo*

```

INICIO
    x : REAL
    x := 5.8
    x := x * 2
FIN
  
```

## Estructuras de control condicionales (o selectivas)

Las estructuras de control condicionales permiten evaluar una expresión lógica (condición que puede ser verdadera o falsa) y, dependiendo del resultado, se realiza uno u otro flujo de instrucciones. Estas estructuras son mutuamente excluyentes (o se ejecuta una acción o se ejecuta la otra)

La estructura de control de flujo más simple es la estructura condicional SI, su sintaxis es la siguiente:

```

SI condición ENTONCES
    [Acción]
FIN SI
  
```

Se evalúa la expresión lógica y si se cumple (si la condición es verdadera) se ejecutan las instrucciones del bloque [Acción]. Si no se cumple la condición, se continúa con el flujo normal del programa.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	57/185
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

### Ejemplo

```
INICIO
  a,b: ENTERO
  a := 3
  b := 2
  SI a > b ENTONCES
    ESCRIBIR "a es mayor"
  FIN SI
FIN

// >>> a es mayor
```

NOTA: La línea //>>> valor, indica el resultado que genera el ejemplo.

La estructura condicional completa es SI-DE LO CONTRARIO:

```
SI cond_booleana ENTONCES
  [Acciones SI]
FIN SI
DE LO CONTRARIO
  [Acciones DE LO CONTRARIO]
FIN DE LO CONTRARIO
```

Se evalúa la expresión lógica y si se cumple (si la condición es verdadera) se ejecutan las instrucciones del bloque SI [Acciones SI]. Si no se cumple la condición se ejecutan las instrucciones del bloque DE LO CONTRARIO [Acciones DE LO CONTRARIO]. Al final el pseudocódigo sigue su flujo normal.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	58/185
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

### Ejemplo

INICIO

```
a,b:ENTERO  
a := 3  
b := 5  
SI a > b ENTONCES  
    ESCRIBIR "a es mayor"  
FIN SI  
DE LO CONTRARIO  
    ESCRIBIR "b es mayor"  
FIN DE LO CONTRARIO  
FIN  
  
// >>> b es mayor
```

La estructura condicional SELECCIONAR-CASO valida el valor de la variable que está entre paréntesis y comprueba si es igual al valor que está definido en cada caso. Si la variable no tiene el valor de ningún caso se va a la instrucción por defecto (DEFECTO).

```
SELECCIONAR (variable) EN  
CASO valor1 -> [Acción]  
CASO valor2 -> [Acción]  
CASO valor3 -> [Acción]  
DEFECTO -> [Acción]  
FIN SELECCIONAR
```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	59/185
		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			

### Ejemplo

```

INICIO
  a :ENTERO
  a := 1
  SELECCIONAR (a) EN
    CASO 1 ->
      ESCRIBIR "Iniciar sesión."
    CASO 2 ->
      ESCRIBIR "Registrarse."
    CASO 3 ->
      ESCRIBIR "Salir."
    DEFECTO ->
      ESCRIBIR "Opción inválida."
  FIN SELECCIONAR
FIN

// >>> "Iniciar sesión"

```

## Estructuras de control iterativas o repetitivas

Las estructuras de control de flujo **iterativas o repetitivas** (también llamadas cíclicas) permiten ejecutar una serie de instrucciones mientras se cumpla la expresión lógica. Existen dos tipos de expresiones cíclicas MIENTRAS y HACER- MIENTRAS.

La estructura MIENTRAS (WHILE en inglés) primero valida la condición y si ésta es verdadera procede a ejecutar el bloque de instrucciones de la estructura, de lo contrario rompe el ciclo y continúa el flujo normal del pseudocódigo.

```

MIENTRAS condición ENTONCES
  [Acción]
FIN MIENTRAS

```

El final de la estructura lo determina la etiqueta FIN MIENTRAS.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	60/185
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

### Ejemplo

```
INICIO
    valorInicial,valorFinal:ENTERO
    valorInicial=0
    valorFinal=3
    MIENTRAS valorInicial < valorFinal
        ESCRIBIR valorInicial
        valorInicial := valorInicial + 1
    FIN MIENTRAS
FIN

//>>> 0
//>>> 1
//>>> 2
```

La estructura HACER-MIENTRAS primero ejecuta las instrucciones descritas en la estructura y al final valida la expresión lógica.

```
HACER
    [Acción]
    MIENTRAS condición
```

Si la condición se cumple vuelve a ejecutar las instrucciones de la estructura, de lo contrario rompe el ciclo y sigue el flujo del pseudocódigo. Esta estructura asegura que, por lo menos, se ejecuta una vez el bloque de la estructura, ya que primero ejecuta y después pregunta por la condición.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	61/185
		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			

### Ejemplo

```

INICIO
    valorInicial,valorFinal:ENTERO
    valorInicial=0
    valorFinal=3
    HACER
        ESCRIBIR valorInicial
        valorInicial := valorInicial + 1
        MIENTRAS valorInicial < valorFinal
    FIN

// >>> 0
// >>> 1
// >>> 2

```

### Funciones

Cuando la solución de un problema es muy compleja se suele ocupar el diseño descendente (divide y vencerás). Este diseño implica la división de un problema en varios subprocessos más sencillos que juntos forman la solución completa. A estos subprocessos se les llaman métodos o funciones.

Una función está constituida por un identificador de función (nombre), de cero a n parámetros de entrada y un valor de retorno:

```

INICIO
    FUNC  identificador  (var:TipoDatos,..., var:TipoDatos)  RET:
        TipoDatos
            [Acciones]
    FIN FUNC
FIN

```

El identificador es el nombre con el que llama a la función. Las funciones pueden o no recibir algún(os) parámetro(s) (tipo(s) de dato(s)) como entrada; si la función recibe alguno se debe incluir entre los paréntesis. Todas las funciones pueden regresar un valor al final de su ejecución (el resultado).



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	62/185
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

Todas las estructuras de control de flujo (secuencial, condicional y repetitivas o iterativas) deben ir dentro de alguna función.

### Ejemplo

#### INICIO

```
FUNC principal (vacío) RET: vacío
    a, b, c: ENTERO
    a := 5
    b := 24
    c := sumar(a, b)
    ESCRIBIR c
FIN FUNC
```

FIN

#### INICIO

```
** Función que suma dos números enteros
FUNC sumar (uno:ENTERO, dos: ENTERO) RET: ENTERO
    enteroTres: ENTERO
    enteroTres:= uno + dos
    RET enteroTres
FIN FUNC
```

FIN

// >>> 29

NOTA: Los dos asteriscos (\*\*) dentro de un pseudocódigo se utilizan para hacer un comentario y, por tanto, lo que esté en la misma línea y después de los \*\* no es parte del algoritmo y no se toma en cuenta. Es una buena práctica realizar comentarios sobre una función o sobre un bloque del algoritmo para guiar sobre el funcionamiento del mismo.

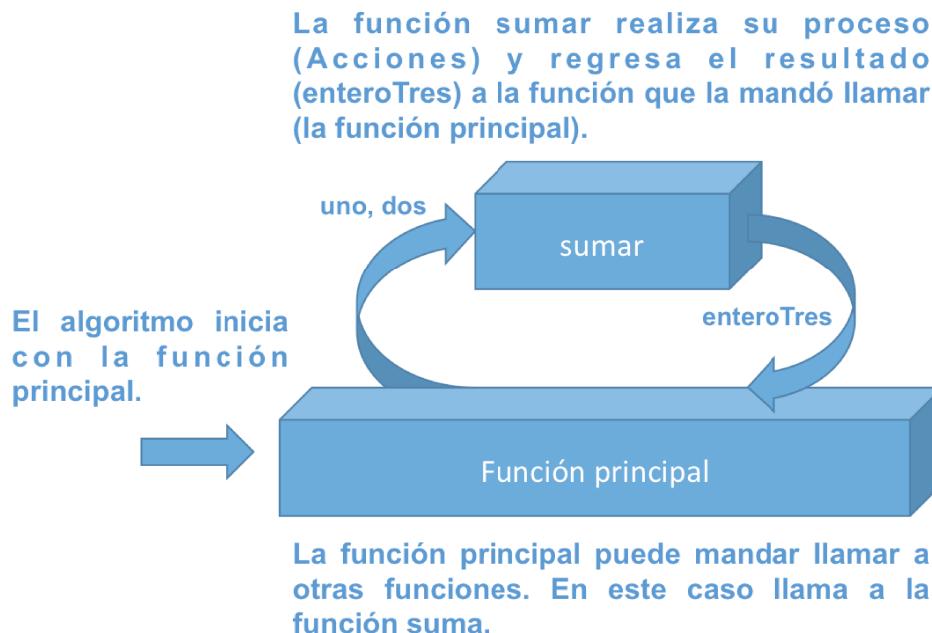
	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	63/185
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería	Área/Departamento: Laboratorio de computación salas A y B		
La impresión de este documento es una copia no controlada			

## Descripción

La primera función que se ejecuta es 'principal', ahí se crean las variables (uno y dos) y, posteriormente, se manda llamar a la función 'sumar'. La función 'sumar' recibe como parámetros dos valores enteros y devuelve como resultado un valor de tipo entero, que es la suma de los valores que se enviaron como parámetro.

Para la función 'principal' los pasos que realiza la función 'sumar' son transparentes, es decir, solo manda a llamar a la función y espera el parámetro de retorno.

La siguiente figura permite analizar el pseudocódigo a través del tiempo. El algoritmo inicia con la función principal, dentro de esta función se hace una llamada a una función externa (sumar). Sumar realiza su proceso (ejecuta su algoritmo) y devuelve un valor a la función principal, la cual sigue su flujo hasta que su estructura secuencial (las instrucciones del pseudocódigo) llega a su fin.



	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	64/185
		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			

## Bibliografía

- Metodología de la programación. Osvaldo Cairó, tercera edición, México D.F., Alfaomega 2005.



- Metodología de la programación a través de pseudocódigo. Miguel Ángel Rodríguez Almeida, primera edición, McGraw Hill





**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	65/185
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 05: Diagramas de flujo



***Elaborado por:***

M.C. Edgar E. García Cano  
Ing. Jorge A. Solano Gálvez

***Revisado por:***

Ing. Laura Sandoval Montaño

***Autorizado por:***

M.C. Alejandro Velázquez Mena



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	66/185
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 05: Diagramas de flujo

### Objetivo:

Elaborar diagramas de flujo que representen soluciones algorítmicas vistas como una serie de acciones que comprendan un proceso.

### Actividades:

- Analizar un problema.
- Crear un algoritmo para resolver el problema.
- Representar el algoritmo en diagrama de flujo.

### Introducción

Un diagrama de flujo es la representación gráfica de un proceso, es decir, muestra gráficamente el flujo de acciones a seguir para cumplir con una tarea específica.

Dentro de las ciencias de la computación, un diagrama de flujo es la representación gráfica de un algoritmo. La correcta construcción de estos diagramas es fundamental para la etapa de codificación, ya que, a partir del diagrama de flujo es posible codificar un programa en algún lenguaje de programación.

### Formas de los diagramas de flujo



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	67/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

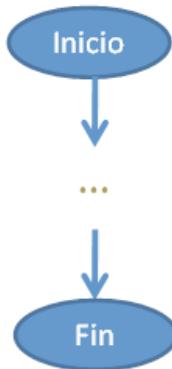
Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

Los diagramas de flujo poseen símbolos que permiten estructurar la solución de un problema de manera gráfica. A continuación, se muestran los elementos que conforman este lenguaje gráfico.

1. Todo diagrama de flujo debe tener un inicio y un fin.



2. Las líneas utilizadas para indicar la dirección del flujo del diagrama deben ser rectas, verticales u horizontales, exclusivamente.



3. Todas las líneas utilizadas para indicar la dirección del flujo del diagrama deben estar conectadas a un símbolo.



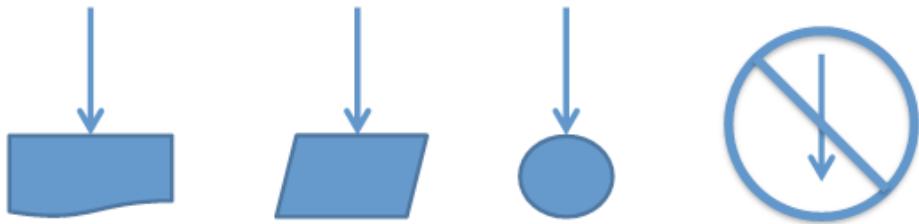
## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	68/185
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



4. El diagrama debe ser construido de arriba hacia abajo (top-down) y de izquierda a derecha (left to right).
5. La notación utilizada en el diagrama de flujo debe ser independiente del lenguaje de programación en el que se va a codificar la solución.
6. Se recomienda poner comentarios que expresen o ayuden a entender un bloque de símbolos.
7. Si la extensión de un diagrama de flujo ocupa más de una página, es necesario utilizar y numerar los símbolos adecuados.



## Manual de prácticas del Laboratorio de Programación básica

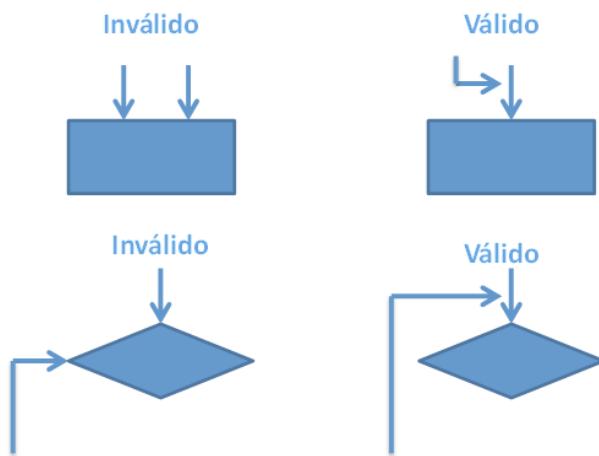
Código:	MADO-18
Versión:	01
Página	69/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

8. A cada símbolo solo le puede llegar una línea de dirección de flujo.



9. Notación de camello. Para nombrar variables y nombres de funciones se debe hacer uso de la notación de camello.

Los diagramas de flujo poseen símbolos que permiten estructurar la solución de un problema de manera gráfica. Por tanto, es fundamental conocer los elementos que conforman este lenguaje gráfico.



Representa el inicio o el fin del diagrama de flujo.

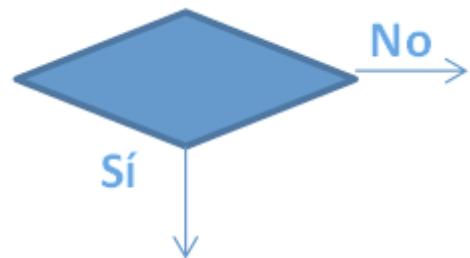
	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	70/185
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Datos de entrada. Expresa lectura de datos.



Proceso. En su interior se expresan asignaciones u operaciones.

Decisión. Valida una condición y toma uno u otro camino.



	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	71/185
		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			



Escritura. Impresión del o lo(s) resultado(s).



Dirección de flujo del diagrama.



Conexión dentro de la misma página.



Conexión entre diferentes páginas.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	72/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

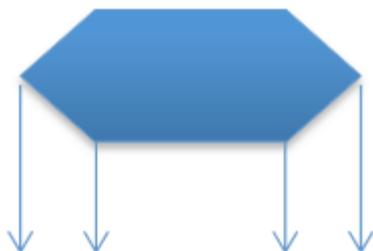
Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada



Módulo de un problema. Llamada a otros módulos o funciones.



Decisión múltiple. Almacena un selector que determina la rama por la que sigue el flujo.



## Manual de prácticas del Laboratorio de Programación básica

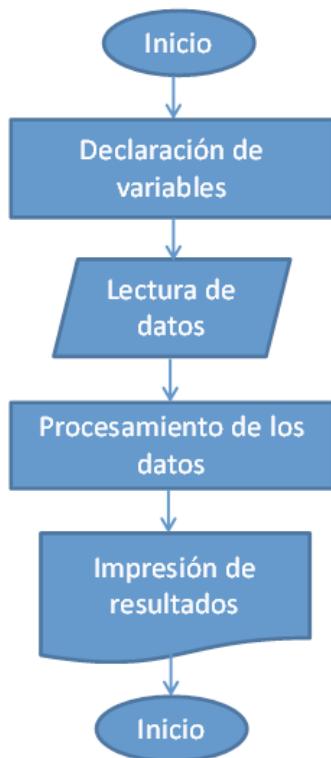
Código:	MADO-18
Versión:	01
Página	73/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

El diagrama de flujo para construir un diagrama de flujo es el siguiente:



	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	74/185
		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			

## Estructuras de control de flujo

Las estructuras de control de flujo permiten la ejecución condicional y la repetición de un conjunto de instrucciones.

Existen 3 estructuras de control: secuencial, condicional y repetitivas o iterativas.

### Estructura de control secuencial

Las estructuras de control secuenciales son las sentencias o declaraciones que se realizan una a continuación de otra en el orden en el que están escritas.

#### *Ejemplo*

```
x: REAL
x ← 5.8
x ← x*2
```

### Estructuras de control condicionales (o selectivas)

Las estructuras de control condicionales permiten evaluar una expresión lógica (condición que puede ser verdadera o falsa) y, dependiendo del resultado, se realiza uno u otro flujo de instrucciones. Estas estructuras son mutuamente excluyentes (o se ejecuta una acción o se ejecuta la otra).

La estructura de control de flujo más simple es la estructura condicional SI (IF), su sintaxis es la siguiente:



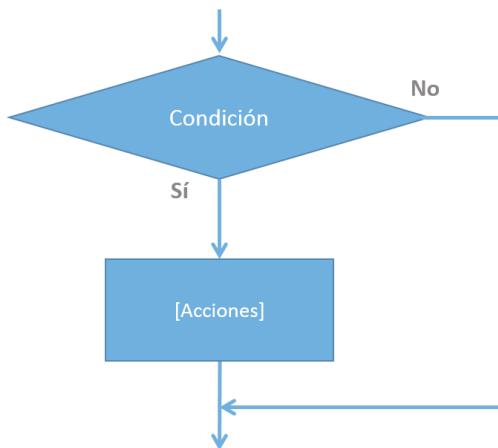
## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	75/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

Facultad de Ingeniería

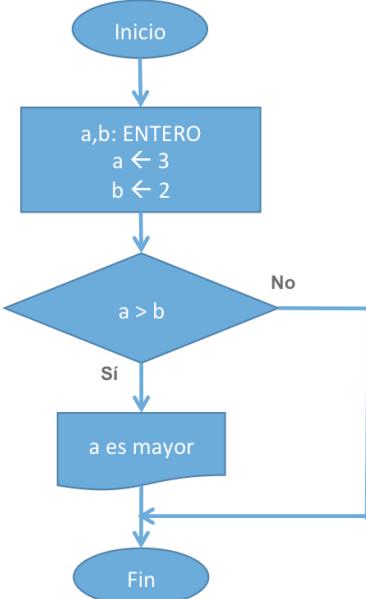
Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada



Se evalúa la expresión lógica y si se cumple (si la condición es verdadera) se ejecutan las instrucciones del bloque [Acciones]. Si no se cumple la condición, se continúa con el flujo normal del programa.

### Ejemplo



// >>> a es mayor



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	76/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

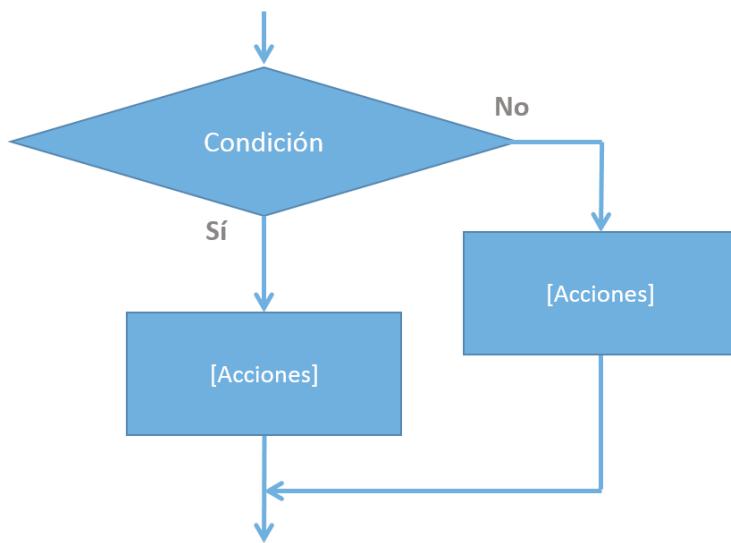
Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

**NOTA:** La línea //>>> valor, indica el resultado que genera el ejemplo.

La estructura condicional completa es SI-DE LO CONTRARIO (IF-ELSE):



Se valúa la expresión lógica y si se cumple (si la condición es verdadera) se ejecutan las instrucciones del bloque Sí. Si no se cumple la condición se ejecutan las instrucciones del bloque No. Al final el programa sigue su flujo normal.



## Manual de prácticas del Laboratorio de Programación básica

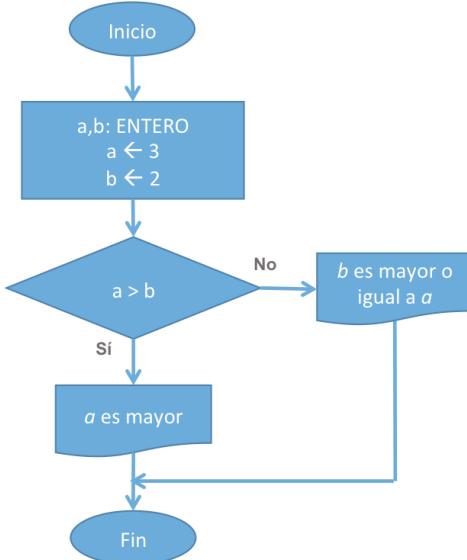
Código:	MADO-18
Versión:	01
Página	77/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

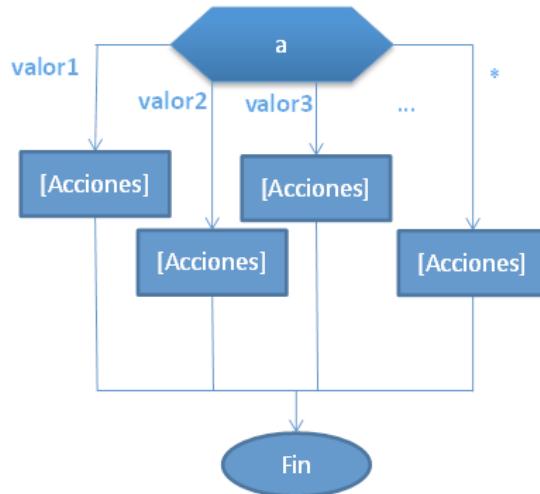
La impresión de este documento es una copia no controlada

### Ejemplo



// >>> b es mayor

La estructura condicional SELECCIONAR-CASO valida el valor de la variable que está en el hexágono y comprueba si es igual al valor que está definido en cada caso (Líneas queemanan del hexágono). Si la variable no tiene el valor de algún caso se va a la instrucción por defecto (\*).





## Manual de prácticas del Laboratorio de Programación básica

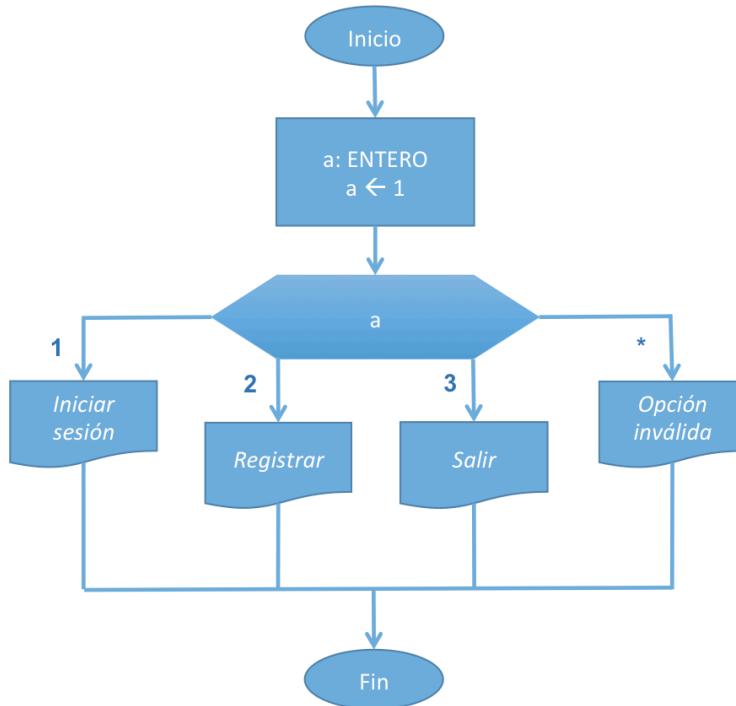
Código:	MADO-18
Versión:	01
Página	78/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

### Ejemplo



// >>> "Iniciar sesión"

### Estructuras de control iterativas o repetitivas

Las estructuras de control de flujo **iterativas o repetitivas** (también llamadas cíclicas) permiten ejecutar una serie de instrucciones mientras se cumpla la expresión lógica. Existen dos tipos de expresiones cíclicas MIENTRAS y HACER- MIENTRAS.

La estructura MIENTRAS primero valida la condición y si ésta es verdadera procede a ejecutar el bloque de instrucciones de la estructura, de lo contrario rompe el ciclo y continúa el flujo normal del programa.



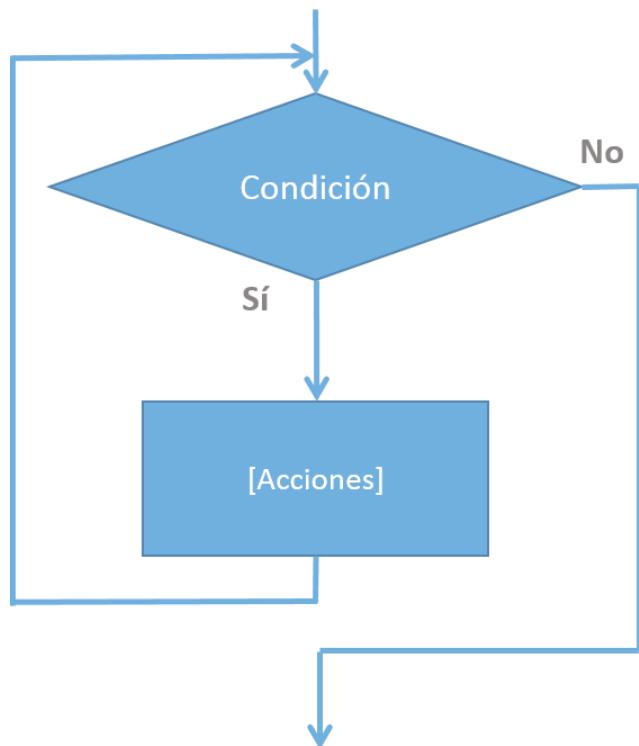
## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	79/185
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





## Manual de prácticas del Laboratorio de Programación básica

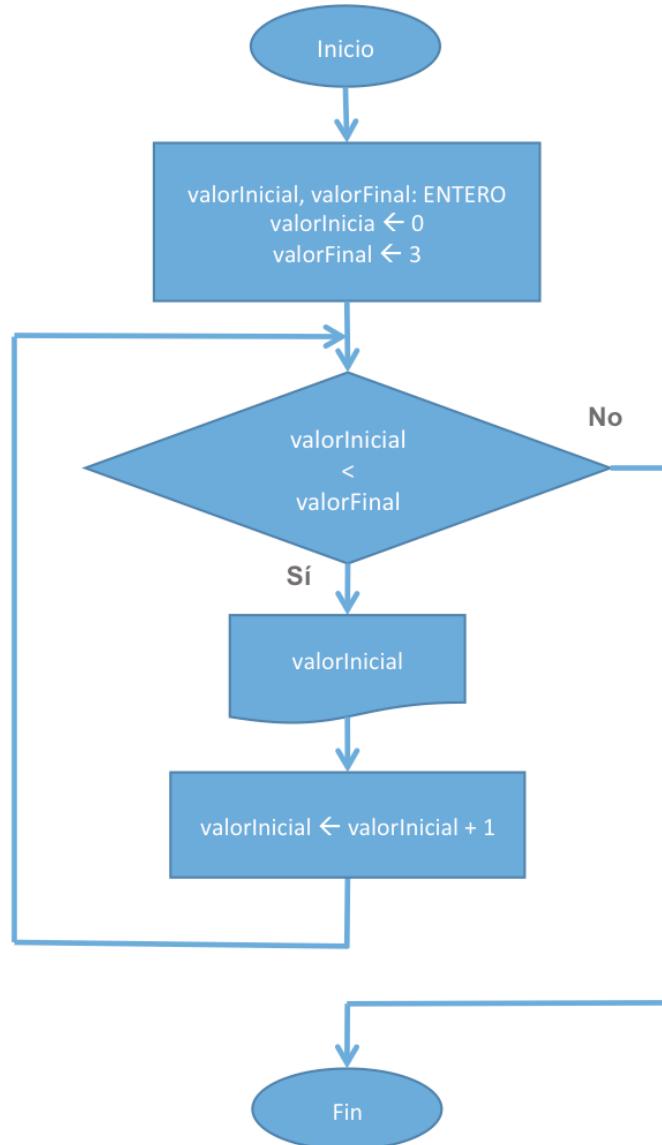
Código:	MADO-18
Versión:	01
Página	80/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

### Ejemplo



```
//>>> 0  
//>>> 1  
//>>> 2
```



## Manual de prácticas del Laboratorio de Programación básica

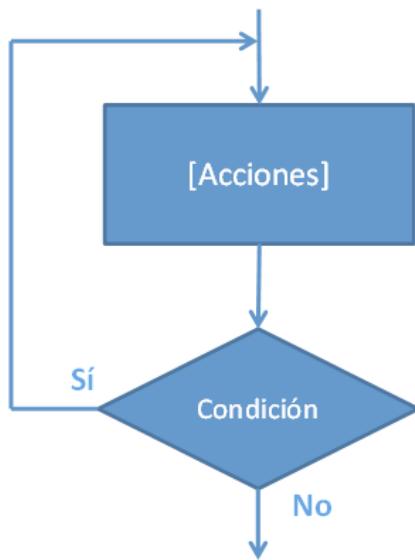
Código:	MADO-18
Versión:	01
Página	81/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

La estructura HACER-MIENTRAS primero ejecuta las instrucciones descritas en la estructura y al final valida la expresión lógica.



Si la condición se cumple vuelve a ejecutar las instrucciones de la estructura, de lo contrario rompe el ciclo y sigue el flujo del algoritmo. Esta estructura asegura que, por lo menos, se ejecuta una vez el bloque de la estructura, ya que primero ejecuta y después pregunta por la condición.

### Ejemplo



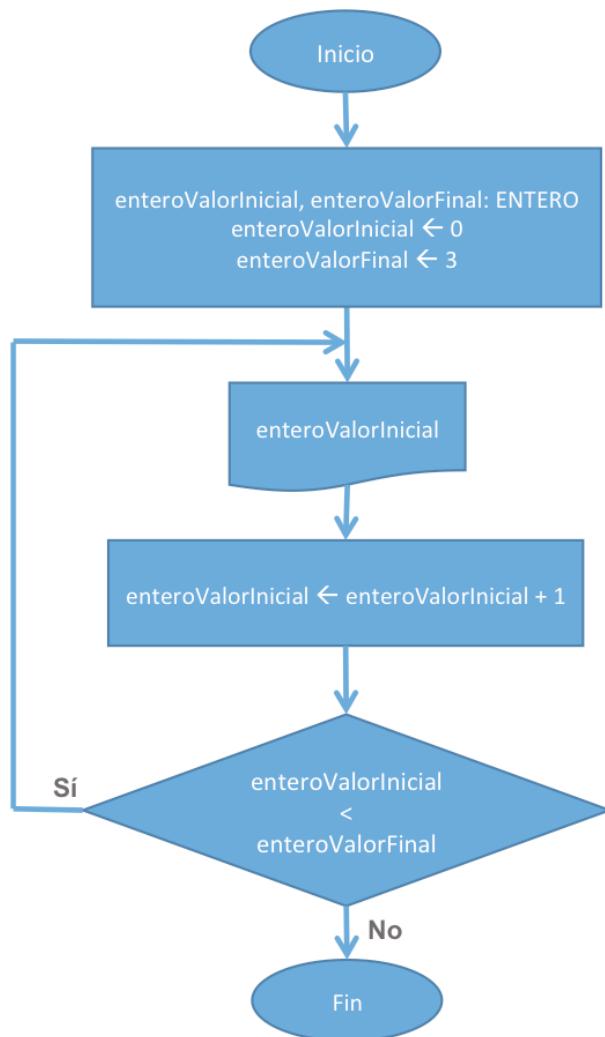
## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	82/185
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



```
// >>> 0
// >>> 1
// >>> 2
```

## Funciones



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	83/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

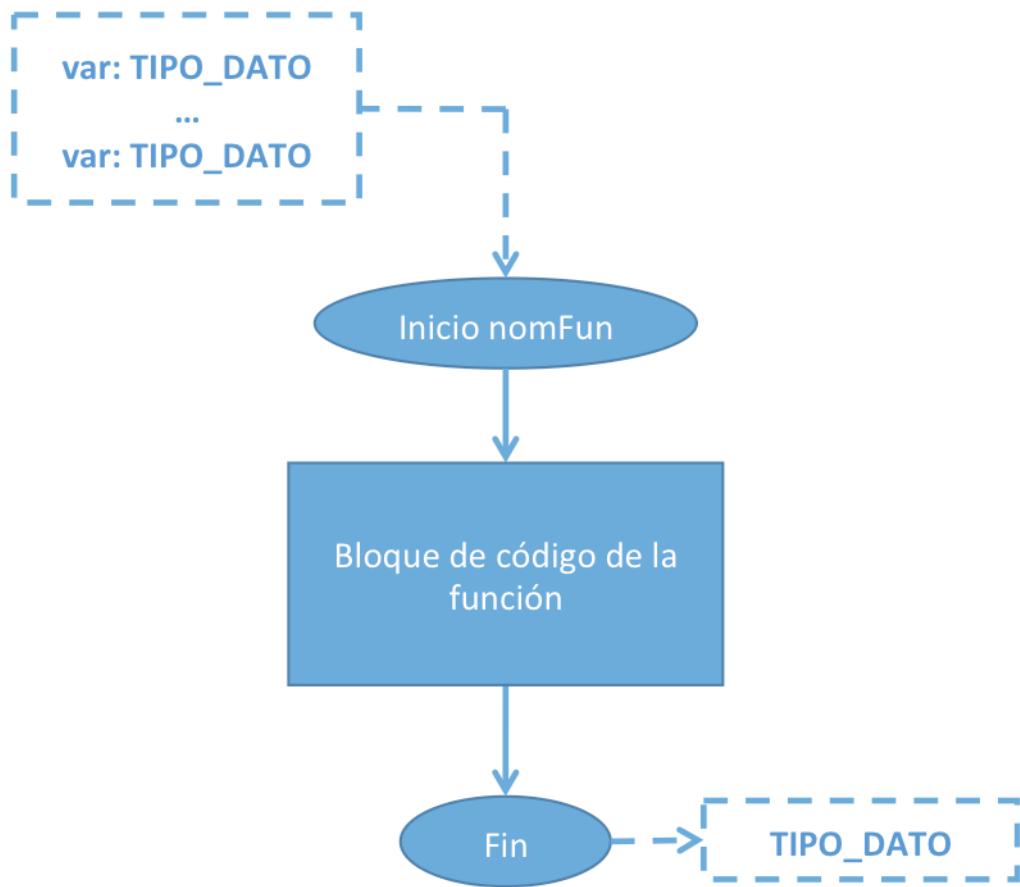
Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

Cuando la solución de un problema es muy compleja se suele ocupar el diseño descendente (divide y vencerás). Este diseño implica la división de un problema en varios subprocessos más sencillos que juntos forman la solución completa. A estos subprocessos se les llaman módulos o funciones.

Una función está constituida por un identificador de función (nombre), de cero a n parámetros de entrada y un valor de retorno:



nomFun es el nombre con el que llama a la función. Las funciones pueden o no recibir algún parámetro (tipo de dato) como entrada, si la función recibe alguno se debe incluir en el recuadro inicial (el que apunta al nombre de la función). Todas las funciones pueden regresar un valor al final de su ejecución (un resultado) para ello se debe definir el dominio del conjunto de salida (tipo de dato).



## Manual de prácticas del Laboratorio de Programación básica

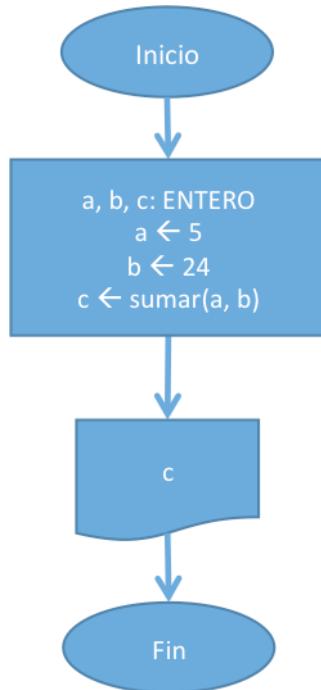
Código:	MADO-18
Versión:	01
Página	84/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

### Ejemplo





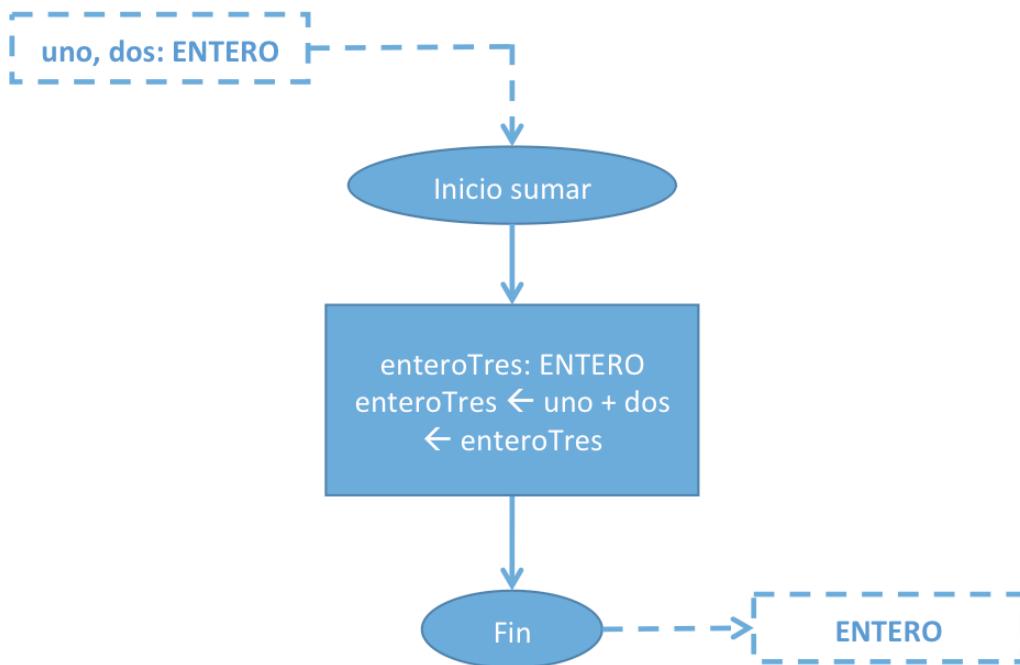
## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	85/185
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



// >>> 29

### Descripción

La primera función que se ejecuta es 'principal', ahí se crean las variables (uno y dos) y, posteriormente, se manda llamar a la función 'sumar'. La función 'sumar' recibe como parámetros dos valores enteros y devuelve como resultado un valor de tipo entero, que es la suma de los valores que se enviaron como parámetro.

Para la función 'principal' los pasos que realiza la función 'sumar' son transparentes, es decir, solo manda a llamar a la función y espera el parámetro de retorno.

La siguiente figura permite analizar la función a través del tiempo. El algoritmo inicia con la función principal, dentro de esta función se hace una llamada a una función externa (sumar). Sumar realiza su proceso (ejecuta su algoritmo) y devuelve un valor a la función principal, la cual sigue su flujo hasta que su estructura secuencial llega a su fin.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	86/185
Sección ISO	8.3
Fecha de emisión	20 de enero de 2017

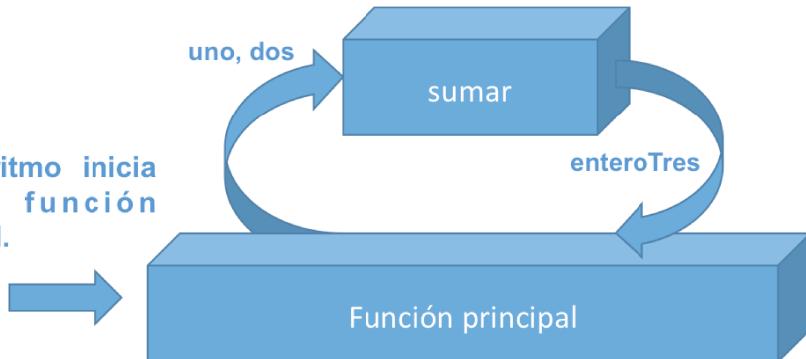
Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

La función sumar realiza su proceso (Acciones) y regresa el resultado (enteroTres) a la función que la mandó llamar (la función principal).

El algoritmo inicia con la función principal.



La función principal puede mandar llamar a otras funciones. En este caso llama a la función suma.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	87/185
		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			

## Bibliografía

- Metodología de la programación. Osvaldo Cairó, tercera edición, México D.F., Alfaomega 2005.



- Metodología de la programación a través de pseudocódigo. Miguel Ángel Rodríguez Almeida, primera edición, McGraw Hill





**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	88/185
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 06: GNU/Linux



---

***Elaborado por:***

Ing. Jorge A. Solano Gálvez  
M.C. Edgar E. García Cano

***Revisado por:***

M.C.

***Autorizado por:***

M.C. Alejandro Velázquez Mena

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	89/185
		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 06: GNU/Linux**

### **Objetivo:**

Explorar un sistema operativo GNU/Linux.

### **Actividades:**

- Conocer los comandos básicos en GNU/Linux.
- Crear y editar archivos de texto a través del editor vi.

### **Introducción**

Linux es un sistema operativo tipo Unix de libre distribución para computadoras personales, servidores y estaciones de trabajo.

El sistema está conformado por el núcleo (kernel) y un gran número de programas y bibliotecas. Muchos programas y bibliotecas han sido posibles gracias al proyecto GNU, por lo mismo, se conoce a este sistema operativo como GNU/Linux.

### **Software libre**

Un software libre es aquel que se puede adquirir de manera gratuita, es decir, no se tiene que pagar algún tipo de licencia a alguna casa desarrolladora de software por el uso del mismo.

Además, que un software sea libre implica también que el software viene acompañado del código fuente, es decir, se pueden realizar cambios en el funcionamiento del sistema si así se desea.

Linux se distribuye bajo la Licencia Pública General de GNU por lo tanto, el código fuente tiene que estar siempre accesible y cualquier modificación o trabajo derivado tiene que tener esta licencia.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	90/185
		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			

## Licencia GNU

La Licencia Pública General de GNU o GNU General Public License (GNU GPL) es una licencia creada por la Free Software Foundation en 1989 y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

## Kernel de GNU/Linux

El kernel ó núcleo de linux se puede definir como el corazón del sistema operativo. Es el encargado de que el software y el hardware del equipo se puedan comunicar.

Entre las funciones más importantes del kernel están:

- Administración de la memoria para todos los programas y procesos en ejecución.
- Administración del tiempo de procesador que los programas y procesos en ejecución utilizan.
- Administra el acceso a los periféricos y/o elementos de la computadora de una manera cómoda.



Figura 1: Capas que componen al sistema operativo GNU/Linux.

## Interfaz de línea de comandos (CLI) o shell de GNU/Linux



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	91/185
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

El Shell de GNU/Linux permite introducir órdenes (comandos) y ejecutar programas en el sistema operativo. Todas las órdenes de UNIX/Linux son programas que están almacenados en el sistema de archivos y a los que llamamos comandos, por lo tanto, todo en GNU/Linux se puede controlar mediante comandos.

### Comandos básicos

La sintaxis que siguen los comandos es la siguiente:

comando [-opciones] [argumentos]

Esto es, el nombre del comando, seguido de algunas banderas (opciones) para modificar la ejecución del mismo y, al final, se puede incluir un argumento (ruta, ubicación, archivo, etcétera) dependiendo del comando. Tanto las opciones como los argumentos son opcionales.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	92/185
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

### Ejemplo (comando ls)

El comando ls permite listar los elementos que existen en alguna ubicación del sistema operativo. Por defecto lista los elementos que existen en la ubicación actual:

`ls`

El comando ls realiza acciones distintas dependiendo de las banderas que utilice, por ejemplo, si se utiliza la opción l se genera un listado largo de la ubicación actual:

`ls -l`

Es posible listar los elementos que existen en cualquier ubicación del sistema operativo, para ello hay que ejecutar el comando especificando como argumento la ubicación donde se desean listar los elementos:

`ls /home`

Tanto las opciones como los argumentos se pueden combinar para generar una ejecución más específica:

`ls -l /home`

GNU/Linux proporciona posee el comando man, el cual permite visualizar la descripción de cualquier comando así como la manera en la que se puede utilizar.

`man ls`

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	93/185
		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			

### **Ejemplo (comando touch)**

El comando touch permite crear un archivo de texto, su sintaxis es la siguiente:

```
touch nombre_archivo[.ext]
```

En GNU/Linux no es necesario agregar una extensión al archivo creado, sin embargo, es recomendable hacerlo para poder identificar el tipo de archivo creado.

### **Ejemplo (comando mkdir)**

El comando mkdir permite crear una carpeta, su sintaxis es la siguiente:

```
mkdir nombre_carpeta
```

### **Ejemplo (comando cd)**

El comando cd permite acceder a una carpeta, su sintaxis es la siguiente:

```
cd nombre_carpeta
```

### **Ejemplo (comando pwd)**

El comando pwd permite conocer la ruta ubicación actual, su sintaxis es la siguiente:

```
pwd
```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	94/185
		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			

### Ejemplo (comando find)

El comando find permite buscar un elemento dentro del sistema, su sintaxis es la siguiente:

```
find . -name cadena_buscar
```

Al comando find hay que indicar en qué parte del sistema va a iniciar la búsqueda. En el ejemplo anterior la búsqueda se inicia en la posición actual. Además, utilizando la bandera `-name` permite determinar la cadena a buscar.

### Ejemplo (comando clear)

El comando clear permite limpiar la consola, su sintaxis es la siguiente:

```
clear
```

### Ejemplo (comando cp)

El comando cp permite copiar un archivo, su sintaxis es la siguiente:

```
cp archivo_origen archivo_destino
```

### Ejemplo (comando mv)

El comando mv mover un archivo de un lugar a otro, su sintaxis es la siguiente:

```
mv ubicación_actual/archivo ubicación_destino
```

El comando mueve el archivo desde su ubicación actual hacia la ubicación deseada. Este comando también puede ser usado para cambiar el nombre de un archivo.

### Ejemplo (comando rm)

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	95/185
		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			

El comando rm permite eliminar un archivo o un directorio, su sintaxis es la siguiente:

```
rm nombre_archivo
rm nombre_carpeta
```

Cuando la carpeta que se desea contiene información se debe utilizar la bandera –f para forzar la eliminación. Si la carpeta contiene otras carpetas, se debe utilizar la opción –r, para realizar la eliminación recursiva.

## Editor Visual Interface de GNU/Linux

El editor vi (visual interface) es un editor de texto de pantalla completa que maneja en memoria el texto entero de un archivo.

Es el editor clásico de UNIX; está en todas las versiones. Puede usarse en cualquier tipo de terminal con un mínimo de teclas.

El editor vi posee tres modos operativos:

- Modo comando: En este modo se ejecuta el editor cada vez que abre un archivo. Permite introducir comandos.
- Modo texto o inserción: Este modo permite la capturar caracteres dentro del documento.
- Modo última línea o reemplazo: Este modo permite escribir comandos en la última línea (al final de la pantalla).

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	96/185
		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			

## Modo de comando

i	inserta texto a la izquierda del cursor
a	agrega texto a la derecha del cursor
x	borra el carácter bajo el cursor
o	añadir una línea en blanco
J	borrar el final de línea (une dos líneas)
u	deshacer la última edición
dd	borra una línea
w	avanza una palabra
b	va al inicio de la palabra actual
e	va al fin de la palabra actual
h o flecha izquierda	mueve el cursor un carácter a la izquierda
j o flecha abajo	mueve el cursor una línea hacia abajo
k o flecha arriba	mueve el cursor una línea hacia arriba
l o flecha derecha	mueve el cursor un carácter a la derecha
ESC	vuelve a modo comando

## Modo última línea o reemplazo

:q	sale del editor
:w	guarda el archivo
:w nombreArchivo	guarda el archivo como "nombreArchivo"
:w! nombreArchivo	forza el guardado del archivo
:wq	guarda y cierra el archivo
:x	guarda y cierra el archivo
:q!	cierra el archivo sin guardar cambios
:/cadena	busca cadena hacia adelante
?cadena	busca la cadena hacia atrás
:set number	agrega número a las líneas
:set nonumber	quita el número de las líneas
:set list	muestra caracteres ocultos
:set nolist	deshabilita set list
:s/lo que busco/lo que reemplazo/c	reemplazar y confirmar
:s/lo que busco/lo que reemplazo/g	reemplazar una línea
:%s/lo que busco/lo que remplazo/g	reemplazar todo documento
:1,\$s/lo que busco/lo que remplazo/g	reemplazar todo documento



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	97/185
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

El editor vi permite crear cualquier tipo de archivo de texto plano: sql, txt, c, java, c++, php, html, jsp, sh, etc.

Además, permite ejecutar comandos propios de linux dentro del modo última línea. La sintaxis para ejecutar un comando de Linux dentro del editor es la siguiente:

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	98/185
		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			

### Ejemplo (vi)

vi permite crear y editar archivos de texto, su sintaxis es la siguiente:

```
vi nombre_archivo[.ext]
```

Por ejemplo, se puede crear un archivo de fortran de la siguiente manera\_ :

```
vi holaMundo.f
```

La instrucción anterior abre el editor vi con el archivo holaMundo.f.



"fortran.f" [New File] 0,0-1 All



# Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	99/185
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

Por defecto, vi se abre en modo comando, para empezar a ingresar datos hay que cambiar a modo texto, para ello se puede presionar la tecla i para insertar caracteres:

En este modo se puede teclear el programa:

```
program holaMundo
    ~ — jrgsln@rha:~/pb — ssh 132.248.59.5 -l jrgsln
write(*,*) 'HOLA MUNDO!'
stop
end
```

-- INSERT -- 5,11 All



# Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	100/185
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 guardar los cambios se debe regresar al modo comando con la tecla ESC y en modo de última línea teclear :wq para guardar y cerrar el archivo.

```
program holaMundo  
write(*,* ) '¡Hola mundo!'  
stop  
end
```

Los pasos anteriores se deben realizar para generar el código fuente de un programa. A esta etapa se le conoce como la etapa de edición.

Una vez creado el código fuente se puede entrar a la etapa de compilación. Para compilar un programa en FORTRAN se utiliza el comando **gfortran**, su sintaxis es la siguiente:

**gfortran nombre\_archivo.f**

El comando gfortran intenta compilar el programa y, si la compilación es exitosa, se genera el archivo ejecutable con el nombre a.out. Es recomendable generar un archivo ejecutable con el mismo nombre que el código fuente, para ello se puede utilizar la bandera -o. El archivo de salida puede tener extensión .out, .exe o nombrarse sin extensión:

```
gfortran nombre_archivo.f -o nombre_archivo.out  
gfortran nombre_archivo.f -o nombre_archivo.exe  
qfortran nombre_archivo.f -o nombre_archivo
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	101/185
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

Si la etapa de compilación es exitosa, se puede pasar a la etapa de ejecución del programa ejecutable de la siguiente manera:

```
./nombre_archivo.out  
./nombre_archivo.exe  
./nombre_archivo
```

## Bibliografía

- Óscar Vicente Huguet Soriano, Sonia Doménech Gómez. Introducción a Linux. [Figura 1]. Consulta: junio de 2015. Disponible en: [http://mural.uv.es/oshuso/81\\_introduccin\\_a\\_linux.html](http://mural.uv.es/oshuso/81_introduccin_a_linux.html)



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	102/185
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	103/185
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*:

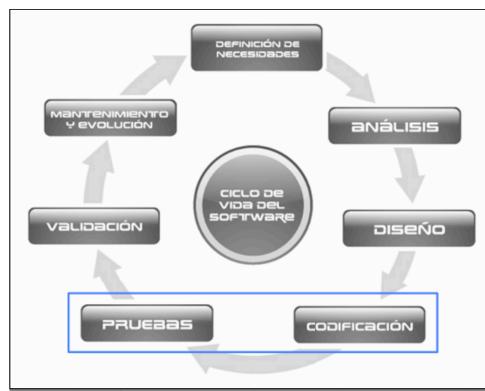


Figura 1: Ciclo de vida del software, resaltando las etapas de *codificación* y *pruebas*, las cuales se cubrirán en esta práctica.

### Lenguaje de programación FORTRAN

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	104/185
		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			

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).

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

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	105/185
		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			

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.
 *
 * 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



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	106/185
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

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	107/185
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:

tipoDeDatos 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.

tipoDeDatos identificador1, identificador2

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	108/185
		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 guion 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.

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

```
parameter (NombreDeVariable = valor)
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	109/185
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 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 carácter
    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
```

*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
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	110/185
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

*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 carácter
    character caracterA

* Variable de tipo lógica
    logical variableLogica

! Asignar un valor a la variable caracterA
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
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	111/185
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 (*,*) '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 carácter:'
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
```

## 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
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	112/185
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

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 horizontal (por columnas)
T	
/	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



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	113/185
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

	reales.
<b>m</b>	Número mínimo de dígitos.
<b>n</b>	Número de espacios saltados
<b>r</b>	Factor de repetición: número de veces que se usa un descriptor o grupo de descriptores.
<b>w</b>	Anchura del campo: Número de caracteres de entrada/salida

La sintaxis para especificar un formato es la siguiente:

```
write (*, et)
et format (descriptores)
```

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.

```
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)
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	114/185
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

```
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
```

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
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	115/185
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

```
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
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:

$$w \geq d+7$$



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	116/185
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

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)
```

```
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



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	117/185
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 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  
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
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	118/185
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

```
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
```

Descriptor de formato para datos de tipo carácter (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 = 'holo'

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

c Sin especificar w
write (*,20) b
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	119/185
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(A)

stop  
end program

## Operadores

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

<i>Operando</i>	<i>Operación</i>	<i>Uso</i>	<i>Resultado</i>
<i>r</i>			<i>o</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

## 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>dble</i>	Convierte un entero a real de doble precisión
<i>ichar</i>	Convierte un carácter a entero
<i>char</i>	Convierte un entero al carácter equivalente en ASCII

### Ejemplo:

```
program conversion
    integer cinco, dos
    double precision resImplicito, resExplicito
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	120/185
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 (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 = dble(cinco)/dble(dos)

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

stop
end
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	121/185
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	122/185
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.

Operador <i>r</i>	Operación	Uso	Resultado
.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.

Operador <i>r</i>	Operación <i>n</i>	Uso
.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

## Código expresiones lógicas



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	123/185
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 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 Languaje 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/](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>



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	124/185
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 08: Estructuras de selección



***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

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	125/185
		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 08: Estructuras de selección

### **Objetivo:**

Elaborar programas en lenguaje FORTRAN que incluyan las estructuras de selección para la resolución de problemas básicos.

### **Actividades:**

- Utilizar las estructuras *if* e *if-else*.
- Utilizar los tipos de datos y expresiones lógicas que se pueden manejar en las estructuras de selección.

### **Introducción**

Las estructuras de control de flujo en un lenguaje especifican el orden en que se realiza el procesamiento de datos.

Las estructuras de selección (o condicionales) permiten realizar una u otra acción con base en una expresión lógica. Las acciones posibles por realizar son mutuamente excluyentes, es decir, solo se puede ejecutar una a la vez dentro de toda la estructura.

Lenguaje FORTRAN posee dos estructuras de selección: la estructura *if-else* y la estructura *select-case*.

### **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.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	126/185
		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			

```
/*
 *
 * 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.
 *
 * 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
 *
 */
```

## Estructura de control selectiva if

La estructura de control de flujo más simple es la estructura condicional **if**, su sintaxis es la siguiente:

```
if (expresión_lógica) ! línea de código a ejecutar
```

En esta estructura se evalúa la expresión lógica y, si se cumple (si la condición es verdadera), se ejecuta la instrucción de la línea. Si no se cumple la condición, se continúa con el flujo normal del programa.

Si el bloque de código a ejecutar consta de más de una línea de código la sintaxis es la siguiente:

```
if (expresión_lógica) then
    Bloque de código a ejecutar
endif
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	127/185
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

En este caso, se evalúa la expresión lógica y, si se cumple (si la condición es verdadera), se ejecuta el bloque de código. Si no se cumple la condición, se omite el bloque de código y se continúa con el flujo normal del programa.

**NOTA 1:** La expresión lógica evaluada regresará como resultado un valor lógico. Dentro de las estructuras de control .FALSE. indica que la expresión lógica es falsa y .TRUE. indica que la expresión lógica es verdadera.

Código (estructura de control selectiva if)

```
program ifSimple

c Este programa valida si el número almacenado en la variable 'a'
c es mayor al número almacenado en la variable 'b'

integer a, b
parameter (a = 3, b = 2)

if (a .GT. b) write (*,*) 'a es mayor que b'
write (*,*) 'El programa sigue su flujo ...'

stop
end
```

Código (estructura de control selectiva if)

```
program ifLogical

c Este programa comprueba las condiciones
c .FALSE. -> falso
c .TRUE. -> verdadero

if (.FALSE.) then
    write (*,*) 'Este bloque de código nunca se ejecuta'
    write (*,*) 'porque la condición siempre es falsa'
endif

if(.TRUE.) then
    write (*,*) 'Este bloque de código siempre se ejecuta'
    write (*,*) 'porque la condición siempre es verdadera'
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	128/185
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

**endif**

**stop**  
**end**

### Estructura de control selectiva if-else

La sintaxis de la estructura de control de flujo **if-else** es la siguiente:

```
if (expresión_lógica) then
c   bloque de código a ejecutar
c   si la condición es verdadera
else
c   bloque de código a ejecutar
c   si la condición es falsa
endif
```

Esta estructura evalúa la expresión lógica y si la condición es verdadera se ejecutan las instrucciones del bloque que se encuentra entre las primeras llaves, si la condición es falsa se ejecuta el bloque de código que está después de la palabra reservada 'else'. Al final de que se ejecute uno u otro código (mutuamente excluyente), se continúa con el flujo normal del programa.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	129/185
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 (estructura de control selectiva if-else)

```
program ifElse

c Este programa permite validar si un
c número entero es par o impar.
c El número se lee desde la entrada
c estándar (el teclado).

integer num

write (*,*) 'Ingrese el numero: '
read (*,*) num

if((num/2) .EQ. 0) then
    write (*,*) 'El numero ', num, ' es par'
else
    write (*,*) 'El numero ', num, ' es impar'
endif

stop
end
```

Es posible *anidar* varias estructuras if-else, es decir, dentro de una estructura if-else tener una o varias estructuras if-else. También es posible validar varias expresiones lógicas en las estructuras else, es decir:

```
if (expresión_lógica) then
    bloque de código a ejecutar
    si la condición_lógica es verdadera
    else (expresión_lógica2)
        bloque de código a ejecutar
        si la condición_lógica2 es verdadera
        else
            bloque de código a ejecutar si
            las condiciones anteriores son falsas
        endif
endif
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	130/185
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 (estructura de control selectiva if-else anidada)

```
program anidadoIfElse

c Este programa ordena en forma descendente
c tres valores enteros dados. Los valores
c se leen desde la entrada estándar (el teclado).

integer uno, dos, tres

write (*,*) 'Ingrese 3 numeros'

read (*,*) uno, dos, tres

if(uno .GT. dos) then
    if(dos .GT. tres) then
        write (*,*) uno,'es mayor que',dos,'que es mayor a',tres
    elseif(uno .GT. tres) then
        write (*,*) uno,'es mayor que',tres,'que es mayor a',dos
    else
        write (*,*) tres,'es mayor que',uno,'que es mayor a',dos
    endif
elseif(dos .GT. tres) then
    if(tres .GT. uno) then
        write (*,*) dos,'es mayor que',tres,'que es mayor que', uno
    else
        write (*,*) dos,'es mayor que',uno,'que es mayor a',tres
    endif
else
    write (*,*) tres,'es mayor que',dos,'que es mayor a',uno
endif

stop
end
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	131/185
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

### Estructura de control selectiva select-case

La sintaxis de la estructura select-case es la siguiente:

```
select case (opcion_a_evaluar){  
    case (valor1)  
        Código a ejecutar  
    case (valor2)  
        Código a ejecutar  
        ...  
    case (valorN)  
        Código a ejecutar  
    case default  
        Código a ejecutar
```

La estructura *select-case* evalúa la variable que se encuentra entre paréntesis después de las palabras reservadas *select case* y la compara con los valores constantes que posee cada caso (*case*). Los tipos de datos que puede evaluar esta estructura son enteros, caracteres y lógicos.

Si la opción a evaluar no coincide dentro de algún caso, entonces se ejecuta el bloque por defecto (*default*). El bloque por defecto normalmente se escribe al final de la estructura, pero se puede escribir en cualquier otra parte.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	132/185
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 (estructura de control select-case)

```
program charCase

c Este programa permite elegir una opción del
c menú a partir del carácter ingresado. La
c opción se lee desde la entrada estándar (el teclado).

character op

write (*,*) 'Menú:'
write (*,*) 'Elegir la opción deseada'
write (*,*) 'a) Ingresar'
write (*,*) 'b) Registrarse'
write (*,*) 'c) Salir'
read (*,*) op

select case (op)

case default
  write (*,*) 'Opción no válida'
case ('a')
  write (*,*) 'Se seleccionó Ingresar'
case ('b')
  write (*,*) 'Se seleccionó Registrarse'
case ('c')
  write (*,*) 'Se seleccionó Salir'
end select

stop
end
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	133/185
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 (estructura de control select-case)

```
program intCase

c Este programa permite elegir una opción del menú a
c partir del entero ingresado. La opción se lee desde
c la entrada estándar (el teclado)

    integer op, valor

    parameter (valor = 3)

    write (*,*) 'Menú'
    write (*,*) 'Elegir la opción deseada'
    write (*,*) '1) Ingresar'
    write (*,*) '2) Registrarse'
    write (*,*) '3) Salir'

    read (*,*) op

    select case (op)

        case (valor-2)
            write (*,*) 'Se seleccionó Ingresar'
        case (valor-1)
            write (*,*) 'Se seleccionó Registrar'
        case (valor)
            write (*,*) 'Se seleccionó Salir'
        case default
            write (*,*) 'Opción no válida'

    end select

    stop
end
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	134/185
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 (estructura de control select-case)

```
program logicalCase

c Este programa permite validar una opción a partir del
c valor lógico ingresado. La opción se lee desde
c la entrada estándar (el teclado)

    logical op

    write (*,*) '¿Desea continuar?'
    write (*,*) 'T) Sí'
    write (*,*) 'F) No'

    read (*,*) op

    select case (op)

        case (.TRUE.)
            write (*,*) 'Se seleccionó Sí'
        case (.FALSE.)
            write (*,*) 'Se seleccionó No'
        case default
            write (*,*) 'Opción no válida'
    end select

    stop
end
```

## Bibliografía

- Oracle (2010). Fortran 77 Languaje 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/](http://web.stanford.edu/class/me200c/tutorial_77/)



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	135/185
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 09: Estructuras de repetición



**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

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	136/185
		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 de práctica de estudio 09: Estructuras de repetición**

### **Objetivo:**

Elaborar programas en FORTRAN para la resolución de problemas básicos que incluyan estructuras de repetición.

### **Actividades:**

- Crear programa con las estructuras while y do-while.
- Anidar estructuras de selección dentro de estructuras repetitivas.

### **Introducción**

Las estructuras de repetición son las llamadas estructuras cíclicas, iterativas o de bucles. Permiten ejecutar un conjunto de instrucciones de manera repetida (o cíclica) mientras que la expresión lógica a evaluar se cumpla (sea verdadera).

En lenguaje FORTRAN existen dos estructuras de repetición: *do-while* y *do*. La estructura *do-while* es una estructura repetitiva de propósito general, la estructura *do* se ejecuta un número predeterminado de veces.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	137/185
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

### 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.
 *
 * 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
 *
 */
```

### Estructura de control repetitiva DO WHILE

La estructura repetitiva (o iterativa) *do-while* primero valida la expresión lógica y si ésta se cumple (es verdadera) procede a ejecutar el bloque de instrucciones de la estructura. Si la condición no se cumple se continúa el flujo normal del programa sin ejecutar el bloque de la estructura, es decir, el bloque se puede ejecutar de cero a *ene* veces. Su sintaxis es la siguiente:

```
do while (expresión_lógica)
c   Bloque de código a repetir
c   mientras que la expresión
c   lógica sea verdadera.
end do
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	138/185
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 (estructura de repetición while)

```
program tablaMultiplicar

c Este programa genera la tabla de multiplicar de un número dado.
c El número se lee desde la entrada est\'andar (teclado).

integer num, cont

cont = 0

write (*,*) '----- Tabla de multiplicar -----'
write (*,*) 'Ingrese un n\'umero:'
read (*,*) num

write (*,*) 'La tabla de multiplicar del',num,'es'

do while (cont .LT. 10)
    cont = cont + 1
    write (*,*) num,'x',cont,'=',num*cont
enddo

stop
end
```

Código (estructura de repetición while)

```
program infinito

c El siguiente es un ciclo infinito
c por que la condici\'on siempre es verdadera.

do while (.TRUE.)
    write (*,*) 'Ciclo infinito.'
    write (*,*) 'Para terminar presione ctrl + c'
end do

stop
end
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	139/185
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 (estructura de repetición do while)

```
program promCalif

c Este programa debe obtener el promedio de calificaciones
c ingresadas por el usuario. Las calificaciones se leen
c desde la entrada estándar (teclado).
c La inserción de calificaciones termina cuando el usuario
c presiona una tecla diferente a las letras 'S' o 's'

character op
real*8 sum, calif
integer veces

veces = 0
sum = 0
op = 's'

do while ((op .EQ. 'S').OR.(op .EQ. 's'))
    write (*,*) 'Suma de calificaciones'
    write (*,*) 'Ingrese calificación'
    read (*,*) calif
    veces = veces + 1
    sum = sum + calif

    write (*,*) '¿Desea sumar otra?'
    read (*,*) op
enddo

write (*,*) 'El promedio de las calificaciones es:'
write (*,*) sum/veces

stop
end
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	140/185
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 (estructura de repetición do while)

```
program calculadora

c Este programa genera una calculadora básica

integer op, uno, dos

do while (op .NE. 5)

    write (*,*) '----- Calculadora -----'
    write (*,*) '¿Que desea hacer?'
    write (*,*) '1) Sumar'
    write (*,*) '2) Restar'
    write (*,*) '3) Multiplicar'
    write (*,*) '4) Dividir'
    write (*,*) '5) Salir'

    read (*,*) op

    select case (op)
        case (1)
            write (*,*) 'Sumar'
            write (*,*) 'Introduzca los números a sumar'
            read (*,*) uno, dos
            write (*,*) uno,'+',dos,'=',uno+dos
        case (2)
            write (*,*) 'Restar'
            write (*,*) 'Introduzca los números a restar'
            read (*,*) uno, dos
            write (*,*) uno,'-',dos,'=',uno-dos
        case (3)
            write (*,*) 'Multiplicar'
            write (*,*) 'Introduzca los números a multiplicar'
            read (*,*) uno, dos
            write (*,*) uno,'*',dos,'=',uno*dos
        case (4)
            write (*,*) 'Dividir'
            write (*,*) 'Introduzca los números a dividir'
            read (*,*) uno, dos
            write (*,*) uno,'/',dos,'=',uno/dos
        case (5)
            write (*,*) 'Salir'

    end select
enddo
stop
end
```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	141/185
		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			

## Estructura de control de repetición DO

Lenguaje FORTRAN posee la estructura de repetición *do* la cual permite realizar repeticiones cuando se conoce el número de elementos que se quiere recorrer. La sintaxis que generalmente se usa es la siguiente:

```

do var = inicio, fin[, incremento]

!Bloque de código
!a ejecutar

enddo

```

La estructura do realiza 3 acciones por sí misma antes o después de ejecutar el bloque de código entre llaves. La primera parte es la inicialización, en la cual se define la variable y se inicializa su valor; esta parte solo se ejecuta una vez cuando se ingresa al ciclo. Fin marca el valor final que puede tomar la variable. El incremento indica el valor con el que la variable es modificada después de cada ejecución del ciclo DO, esta parte es opcional, cuando no aparece, su valor por defecto es 1. El incremento puede ser positivo o negativo.

Código (estructura de repetición do)

```

program doSinIncremento

c Este programa debe obtener el promedio de 5 calificaciones
c ingresadas por el usuario. Las calificaciones se leen
c desde la entrada estándar (teclado).

real*8 sum, calif
integer veces

sum = 0

c El incremento es opcional, si no se escribe por defecto es uno
do veces = 1, 5
    write (*,*) 'Suma de calificaciones'
    write (*,*) 'Ingrese la calificación', veces
    read (*,*) calif
    sum = sum + calif
end do

```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	142/185
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 (*,*) 'El promedio de las calificaciones es:'
write (*,*) sum/(veces-1)

stop
end
```

### Código (estructura de repetición do)

```
program doConIncremento

c Este programa debe obtener el promedio de 5 calificaciones
c ingresadas por el usuario. Las calificaciones se leen
c desde la entrada estándar (teclado).

real*8 sum, calif
integer veces, cont

sum = 0
cont = 1

do veces = 1, 10, 2
    write (*,*) 'Suma de calificaciones'
    write (*,*) 'Ingrese la calificación', cont
    read (*,*) calif
    sum = sum + calif
    cont = cont + 1
end do

write (*,*) 'El promedio de las calificaciones es:'
write (*,*) sum/(cont-1)

stop
end
```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	143/185
		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			

## Sentencia EXIT

La sentencia EXIT produce la salida inmediata de un ciclo. Sirve para crear una repetición controlada por una expresión lógica. Con la ayuda de las estructuras DO e IF y la sentencia EXIT se puede crear un ciclo WHILE, de la siguiente manera:

```

do
  !Bloque de código
  !a ejecutar

  IF (expresión_lógica) EXIT

  !Bloque de código
  !a ejecutar
end do

```

La estructura DO genera un ciclo que se ejecuta MIENTRAS que la expresión lógica no se cumpla (sea falsa). Cuando la expresión lógica de la estructura de selección IF se cumple (es verdadera), se ejecuta la sentencia EXIT, cuyo efecto es transferir el control fuera del ciclo DO, a la primera sentencia siguiente a END DO, de manera similar a un ciclo WHILE. La validación de la condición (estructura IF) se puede situar en cualquier parte de la estructura DO.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	144/185
		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 (sentencia exit)

```

PROGRAM dec2bin

c Programa que permite convertir un número en base 10 a base 2.

INTEGER decimal

WRITE (*,*) 'Ingrese un número decimal'
READ (*,*) decimal

DO
  IF (decimal .LE. 0) EXIT
  WRITE (*,*) MOD(decimal,2)
  decimal = decimal / 2
END DO

WRITE (*,*) 'El número se lee de abajo hacia arriba'

STOP
END

```

## Sentencia CYCLE

La sentencia CYCLE detiene la ejecución de la iteración actual y devuelve el control al inicio del ciclo, continuando la ejecución de la iteración siguiente. La sentencia CYCLE se puede usar dentro de cualquier ciclo iterativo:

```

do while (expresión_lógica)
  !Bloque de código
  !a ejecutar

  IF (expresión_lógica) CYCLE

  !Bloque de código
  !a ejecutar
end do

```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	145/185
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 (sentencia cycle)

```
PROGRAM sumaPares
```

```
c Programa que suma los primeros 5 números pares a partir de un número dado.
```

```
INTEGER numero, suma, contador
suma = 0
contador = 1
```

```
WRITE (*,*) 'Ingrese el número inicial'
READ (*,*) numero
```

```
DO WHILE (contador .LE. 5)
    numero = numero + 1
```

```
c si el número es par se devuelve el control al inicio del ciclo,
c es decir, no se ejecutan las líneas que están debajo de la estructura IF
```

```
IF (MOD(numero, 2) .EQ. 1) CYCLE
```

```
    contador = contador + 1
    suma = suma + numero
```

```
END DO
```

```
WRITE (*,*) 'La suma de los números pares es: ', suma
```

```
STOP
END
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	146/185
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

### Estructuras repetitivas con nombre

Se puede asignar un nombre a una estructura agregando 'nombre:' antes de la palabra reservada DO y 'nombre' después del final del ciclo (END DO). Los nombres son opcionales, pero si se usan deben ser en ambas etiquetas, es decir:

```
[nombre:] do
    !Bloque de código
    !a ejecutar

    IF (expresión_lógica) CYCLE [nombre]

    !Bloque de código
    !a ejecutar

    IF (expresión_lógica) EXIT [nombre]

end do [nombre]
```

```
[nombre:] do var = inicio, fin[, incremento]
    !Bloque de código
    !a ejecutar

    IF (expresión_lógica) CYCLE [nombre]

    !Bloque de código
    !a ejecutar

end do [nombre]
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	147/185
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 (ciclos con etiquetas)

### PROGRAM etiquetas

c Programa que genera 3 números consecutivos (1, 2 y 3) y los divide c entre 3 números proporcionados por el usuario. Si el número c ingresado es cero el programa termina su ejecución.

```
REAL dividendo, divisor
INTEGER contadorExterno, contadorInterno
contadorExterno = 1

externo: DO WHILE (contadorExterno .LE. 3)
    dividendo = contadorExterno
    WRITE (*,*) 'Dividendo: ', dividendo
    contadorInterno = 0
    interno: DO
        WRITE (*,*) 'Ingrese el número inicial'
        READ (*,*) divisor

        IF (divisor .EQ. 0) THEN
            WRITE (*,*) 'No se puede dividir entre 0'
            EXIT externo
        ENDIF

        WRITE (*,*) dividendo, '/', divisor, '=' , dividendo/divisor
        contadorInterno = contadorInterno + 1

        IF (contadorInterno .EQ. 2) EXIT interno
    END DO interno
    contadorExterno = contadorExterno + 1
END DO externo

STOP
END
```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	148/185
		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			

## Bibliografía

- Oracle (2010). Fortran 77 Languaje 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/](http://web.stanford.edu/class/me200c/tutorial_77/)



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	149/185
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 10: Arreglos unidimensionales y apuntadores



**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

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	150/185
		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 10: Arreglos unidimensionales y apuntadores

### **Objetivo:**

Elaborar programas en lenguaje FORTRAN para resolver problemas que requieran agrupar conjuntos de datos del mismo tipo en arreglos unidimensionales.

### **Actividades:**

- Crear arreglos unidimensionales.
- Crear apuntadores.

### **Introducción**

Un arreglo es un conjunto de datos contiguos del mismo tipo con un tamaño fijo, definido al momento de crearse. A cada elemento (dato) del arreglo se le asocia una posición particular. Para acceder a los elementos de un arreglo es necesario utilizar un índice.

### **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.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see
 */

```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	151/185
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

```
<http://www.gnu.org/licenses/>.  
*  
* Author: Jorge A. Solano  
*  
*/
```

## Arreglos unidimensionales

Un arreglo unidimensional de  $n$  elementos en la memoria se almacena de la siguiente manera:

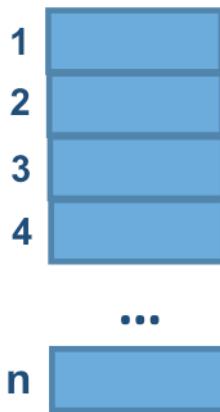


Figura 1. Representación de los  $n$  elementos de un arreglo.

Por defecto, la primera localidad del arreglo corresponde al índice 1 y la última corresponde al índice  $n$ , donde  $n$  es el tamaño del arreglo.

La sintaxis para definir un arreglo en lenguaje FORTRAN es la siguiente:

```
tipoDeDato nombre(tamaño)
```

Donde nombre se refiere al identificador del arreglo, tamaño es un número entero y define el número máximo de elementos que puede contener el arreglo. Un arreglo puede ser de cualquier tipo de dato.

Código (arreglo unidimensional)



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	152/185
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 whileArreglo

c Este programa genera un arreglo unidimensional de
c 5 elementos y accede a cada elemento del arreglo
c a través de un ciclo do while.

integer indice, lista(5)

indice = 1

lista(1) = 10
lista(2) = 8
lista(3) = 5
lista(4) = 8
lista(5) = 7

write (*,*) 'Lista'

do while (indice .LE. 5)
  write (*,*) 'Calificación del alumno',indice,'es',lista(indice)
  indice = indice + 1
enddo

stop
end
```

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	153/185
		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 (estructura de repetición do)

```

program doArreglo

c Este programa genera un arreglo unidimensional de
c 5 elementos y accede a cada elemento del arreglo
c a través de un ciclo do.

integer indice, lista(5)

indice = 1

lista(1) = 10
lista(2) = 8
lista(3) = 5
lista(4) = 8
lista(5) = 7

write (*,*) 'Lista'

do indice = 1, 5
  write (*,*) 'Calificación del alumno',indice,'es',lista(indice)
enddo

stop
end

```

## Apuntadores

Un apuntador es una variable que contiene la dirección de una variable, es decir, hace referencia a la localidad de memoria de otra variable. Debido a que los apuntadores trabajan directamente con la memoria, a través de ellos se accede con rapidez a un dato.

La sintaxis para declarar un apuntador y para asignarle la dirección de memoria de otra variable es, respectivamente:

```

TipoDeDato, pointer :: apuntador
TipoDeDato, target :: variable
apuntador => variable

```

La declaración de una variable apuntador inicia con la palabra reservada pointer, seguida de :: y el identificador del apuntador.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	154/185
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

Los apuntadores solo pueden apuntar a direcciones de memoria del mismo tipo de dato con el que fueron declarados.

### Código (apuntadores)

```
program apuntador
c Este programa crea un apuntador de tipo carácter
c Definición de un apuntador de tipo carácter
c utilizando la palabra reservada pointer
      character, pointer :: ap
c Se utiliza la palabra reservada target para indicar
c que la variable c puede ser apuntada por un pointer
      character, target :: c
      c = 'a'
c se realiza la asignación, ap apunta a la localidad
c de memoria de c
      ap => c
      write (*,*) 'Carácter:', ap
      write (*,*) 'Código ASCII:', ichar(ap)
      stop
end
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	155/185
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

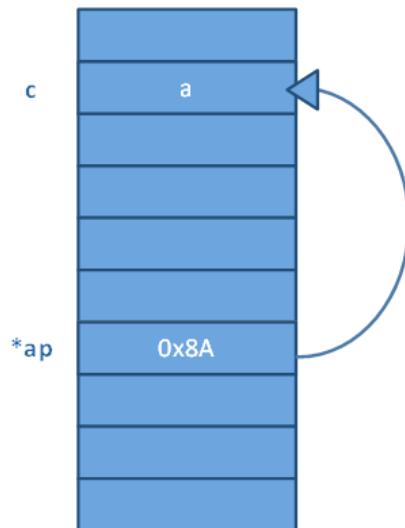


Figura 2. Representación de una variable apuntador en la memoria.

### Código (apuntadores)

```
program apuntador2

c Este programa accede a las localidades de memoria de
c distintas variables a través de un apuntador

c declaración de una variable apuntador
    integer, pointer :: apEnt
c se declara que a, b y el arreglo c pueden ser apuntadas
    integer, target :: a, b, c(10)

a = 5
b = 10
c(1) = 5
c(2) = 4
c(3) = 3
c(4) = 2
c(5) = 1
c(6) = 9
c(7) = 8
c(8) = 7
c(9) = 6
c(10) = 0
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	156/185
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

```
c se asigna la localidad de memoria de 'a' a la variable apuntador
apEnt => a

write (*,*) 'a = ',a
write (*,*) 'b = ',b
write (*,*) 'c(10) = ',c
write (*,*) 'apEnt => a'

b = apEnt
write (*,*) 'b = apEnt -> b =', b

b = apEnt + 1
write (*,*) 'b = apEnt + 1 -> b =', b

apEnt = 0
write (*,*) 'apEnt = 0 -> a =', a

apEnt => c(1)
write (*,*) 'apEnt => c(1) -> apEnt =', apEnt

stop
end
```

Es posible saber si un apuntador está asociado a una variable a través de la función ASSOCIATED. Por otra parte, la función NULLIFY permite desasociar el apuntador de una variable.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	157/185
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 (associated y nullify)

```
program associatedNullify

c Este programa valida si un apuntador tiene referencia
c hacia una variable o no. Tambi?n permite desasociar el
c valor de un apuntador.

    integer, target :: arr(5)
    integer, pointer :: apArr

    arr = (/5, 4, 3, 2, 1/)
c apArr apunta a la primera localidad del arreglo
    apArr => arr(1)

    if (associated(apArr)) then
        write (*,*) 'apArr está apuntando a -> arr(1) = ', apArr
    end if

c se elimina la asociación del apuntador hacia la variable.
    nullify(apArr)
    write (*,*) '¿apArr está asociado?',(associated(apArr))
    write (*,*) 'apArr -> ', apArr

c apArr apunta a la tercera localidad del arreglo
    apArr => arr(3)
    if (associated(apArr)) then
        write (*,*) 'apArr está apuntando a -> arr(3) = ',apArr
    endif

    stop
end
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	158/185
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 (cadenas)

```
program cadena

c Este programa lee una palabra del teclado, de una longitud
c máxima de 20 caracteres
character (len = 20):: palabra

write (*,*) 'Ingrese una palabra: '
read (*,*) palabra
write (*,*) 'La palabra ingresada es: ', palabra

stop
end
```

### Código (arreglos como cadenas)

```
program recorrePalabra

c Este programa lee una palabra de 20 máximo caracteres
c Después imprime en la salida estándar (pantalla) la
c palabra y además imprime el arreglo carácter por
c carácter

character palabra(20)
integer i

write (*,*) 'Ingrese una palabra: '
read (*,*) palabra
write (*,*) 'La palabra ingresada es: ', palabra

do i = 1, 20, 1
    write (*,*) palabra(i)
enddo

stop
end
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	159/185
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

## Bibliografía

- Oracle (2010). Fortran 77 Languaje 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/](http://web.stanford.edu/class/me200c/tutorial_77/)



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	160/185
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 11: Arreglos multidimensionales



**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

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	161/185
		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 11: Arreglos multidimensionales

### **Objetivo:**

Elaborar programas en lenguaje FORTRAN para resolver problemas que requieran agrupar conjuntos de datos del mismo tipo en arreglos multidimensionales.

### **Actividades:**

- Crear arreglos multidimensionales.

### **Introducción**

Un arreglo es un conjunto de datos contiguos del mismo tipo con un tamaño fijo, definido al momento de crearse. Los arreglos pueden ser unidimensionales (como se vio en la práctica anterior) o multidimensionales y se utilizan para hacer más eficiente el código de un programa.

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



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	162/185
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

\*  
\* Author: Jorge A. Solano  
\*  
\*/

## Arreglos multidimensionales

Lenguaje FORTRAN permite crear arreglos de varias dimensiones con la siguiente sintaxis:

tipoDato nombre (tamaño1, tamaño2,...,tamaño7)

o con la siguiente sintaxis:

tipoDato nombre  
dimension nombre (tamaño1, tamaño2,...,tamaño7)  
tipoDato DIMENSION (d1[, d2...]) :: nombre

Donde nombre se refiere al identificador del arreglo, tamaño es un número entero y define el número máximo de elementos que puede contener el arreglo por dimensión. La palabra reservada DIMENSION permite definir el tamaño máximo de las diferentes dimensiones.

Los tipos de dato que puede tolerar un arreglo multidimensional son: entero, real, carácter o complejo.

De manera práctica se puede considerar que la primera dimensión corresponde a los renglones, la segunda a las columnas, la tercera al plano, y así sucesivamente.

Por defecto, los arreglos inician en la posición 1 y, por tanto, las dimensiones se recorren de la posición 1 a la posición d1 para la primera dimensión, de la posición 1 a la posición d2 en la segunda dimensión, y así sucesivamente. Sin embargo, es posible indicarle al compilador de FORTRAN donde inician las dimensiones de la siguiente manera:

tipoDato DIMENSION (inicio\_d1: d1, inicio\_d2:d2, ...)

En este caso, las dimensiones se recorren desde la posición inicio\_d1 hasta la posición d1 para la primera dimensión, de la posición inicio\_d2 hasta la posición d2 para la segunda dimensión y así sucesivamente.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	163/185
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 (arreglos multidimensionales)

```
program arregloBidimensional

c Este programa genera un arreglo bidimensional de
c 3 renglones y 3 columnas y accede a cada elemento
c del arreglo a través de un ciclo do

integer i,j, matriz(3,3)

matriz(1,1) = 1
matriz(1,2) = 2
matriz(1,3) = 3
matriz(2,1) = 4
matriz(2,2) = 5
matriz(2,3) = 6
matriz(3,1) = 7
matriz(3,2) = 8
matriz(3,3) = 9

write (*,*) 'Imprimir matriz'

do i = 1, 3, 1
    do j = 1, 3, 1
        write (*,*) matriz(i,j)
    enddo
enddo

stop
end
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	164/185
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 (arreglos multidimensionales)

```
program restaDeMatrices

c Este programa genera las matrices (A y B) y después
c las resta generando una tercera matriz C

integer i,j,cont
integer, dimension (-4:-1,3) :: matrizA, matrizB
integer matrizC(-4:-1,3)

c se genera la matriz A
cont = 1
do i = -4, -1, 1
    do j = 1, 3, 1
        matrizA(i,j) = cont
        cont = cont + 1
    enddo
    write (*,*) ''
enddo

c se genera la matriz B
do i = -4, -1, 1
    do j = 1, 3, 1
        cont = cont - 1
        matrizB(i,j) = cont
    enddo
    write (*,*) ''
enddo

c se imprimen las matrices generadas
write (*,*) 'Matriz A          Matriz B'
do i = -4, -1, 1
    do j = 1, 3, 1
        write(*,*) matrizA(i,j), ' ', matrizB(i,j)
    enddo
    write (*,*) ''
enddo
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	165/185
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

```
c se genera e imprime la matriz C
write (*,*) 'Matriz C = Matriz A - Matriz B'
do i = -4, -1, 1
    do j = 1, 3, 1
        matriz(i,j) = matrizA(i,j) - matrizB(i,j)
        write (*,*) matrizC(i,j)
    enddo
    write (*,*) ''
enddo

stop
end
```

## Bibliografía

- Oracle (2010). Fortran 77 Languaje 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/](http://web.stanford.edu/class/me200c/tutorial_77/)



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	166/185
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 12: Funciones



**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

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	167/185
		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 12: Funciones

### **Objetivo:**

Elaborar programas en lenguaje FORTRAN que permitan dividir la solución del problema en funciones.

### **Actividades:**

- Distinguir lo que es el prototipo de una función y la implementación o definición de ella.
- Utilizar parámetros tanto en la función principal como en funciones secundarias.

### **Introducción**

Como ya se mencionó, un programa en lenguaje FORTRAN consiste en una o más funciones. FORTRAN permite tener dentro de un archivo fuente varias funciones, esto con el fin de dividir las tareas y que sea más fácil la depuración, la mejora y el entendimiento del código.

### **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.
 *
 * You should have received a copy of the GNU General Public License
 */
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	168/185
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

```
* along with this program. If not, see <http://www.gnu.org/licenses/>.  
*  
* Author: Jorge A. Solano  
*  
*/
```

## Funciones y subrutinas

FORTRAN provee dos maneras de implementar subprogramas: creando funciones o creando subrutinas. Estos subprogramas pueden ser de dos tipos intrínsecos y externos. Los subprogramas intrínsecos son aquellos que provee el compilador, algunos de los más comunes son:

abs	valor absoluto
min	valor mínimo
max	valor máximo
sqrt	raíz cuadrada
sin	seno
cos	coseno
tan	tangente
atan	arco tangente
exp	exponente (natural)
log	logaritmo (natural)

Por otro lado, los subprogramas externos son aquellos escritos por el usuario (o bien pueden formar parte de una biblioteca desarrollados por terceros). Estos son los que se abordarán en este tema.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	169/185
		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			

## Función

La sintaxis básica de una función (también conocida como definición de la función) es la siguiente:

```
valorRetorno nombre (parámetros)
    ! bloque de código de la función
END
```

Por otro lado, el prototipo de una función está compuesto por tres elementos: el nombre de la función, los parámetros que recibe la función y el valor de retorno de la función.

El nombre de la función se refiere al identificador con el cual se ejecutará la función; es recomendable seguir la notación de camello.

Una función puede recibir parámetros de entrada, dichos parámetros se deben definir dentro de los paréntesis de la función y separados por comas, es decir:

```
(nom1, nom2, nom3...)
    tipoDato nom1, nom2, nom3...
```

El tipo de dato puede ser cualquiera de los vistos hasta el momento (entero, lógico, real, carácter, complejo o arreglo). Los parámetros de una función son opcionales.

El valor de retorno de una función indica el tipo de dato que va a regresar la función al terminar el bloque de código de la misma. El valor de retorno puede ser cualquiera de los tipos de datos vistos hasta el momento (entero, lógico, real, carácter, complejo o arreglo).



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	170/185
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 (función)

```
program principal

! Programa que permite calcular los valores de un polinomio
! de grado 2, es decir, del tipo A*x^2 + B*x + C

real sum, x, coef
integer m

write(*,*) 'Escriba el valor de x'
read(*,*) x

sum = 0.0
do 10 m = 0, 2
    write(*,*) 'Escriba el valor del coeficiente de x', m
    read(*,*) coef
    sum = sum + (coef * potencia(x, m))
10 continue
write (*,*) 'El valor del polinomio valuado en ', x, 'es', sum

stop
end

real function potencia(x,m)
real x
integer m
potencia = 1.0
do i = 1, m
    potencia = potencia * x
enddo

return
end
```

La función *principal* hace una llamada a la función *potencia* pasando como parámetro los valores x y m. La función *potencia* ejecuta su bloque de código y cuando termina regresa el valor almacenado en *potencia*, que es de tipo real.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	171/185
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

### Subrutina

Una función en FORTRAN solo puede regresar un valor. Algunas ocasiones es necesario regresar dos o más valores, en este caso se puede hacer uso de las subrutinas de FORTRAN. Su sintaxis básica es la siguiente:

```
SUBROUTINE nombre (parámetros)
    ! bloque de código de la función
END
```

Como se puede observar, las subrutinas no poseen un valor de retorno explícito en la declaración de la misma. Además, para invocar una subrutina se debe hacer uso de la palabra reservada CALL antes del nombre y los parámetros de la misma.

### Código (subrutina)

```
program principal
! Programa que realiza el ordenamiento descendente de 2 números dados
integer m, n
m = 1
n = 2
if (m .LE. n) then
    call intercambiar(m, n)
endif
write(*,*) m, n
stop
end

subroutine intercambiar (a,b)
integer a, b
! Variables locales a la subrutina intercambiar
integer tmp
tmp = a
a = b
b = tmp
return
end
```

La función *principal* manda llamar a la subrutina *intercambiar* pasando como parámetros las variables m y n. La función *intercambiar* recibe como parámetros los valores y los guarda en las variables a y b, los cuales intercambia dentro del bloque de código de la



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	172/185
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

subrutina. Cuando la subrutina llega a su fin, los valores que se intercambiaron son, en realidad, los que envió la función principal y, por ende, se ven afectados directamente, es decir, los valores que se intercambiaron fueron m y n.

Código (subrutina)

```
program principal

integer ene
parameter (ene = 5)
real equis(ene), ye(ene), escalar
parameter (escalar = 4)
do 10 i = 1, ene
    equis(i) = i * 2
    ye(i) = i * 1
10 continue

call saxpy(ene, escalar, equis, ye)

do 11 i = 1, ene
    write(*,*) ye(i)
11 continue

stop
end

subroutine saxpy (a,b)
! La operación SAXPY se calcula de la siguiente manera:
! y = alfa*x + y,
! donde 'x' y 'y' son vectores de longitud n.
    integer n, i
    real alfa, x(*), y(*)
    do 10 i = 1, n
        y(i) = alfa*x(i) + y(i)
10    continue
    return
end
```

## Ámbito o alcance de las variables

Las variables declaradas dentro de un programa tienen un tiempo de vida que depende de la posición donde se declaren, es decir, las variables en FORTRAN sólo existen dentro de



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	173/185
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

las funciones o subrutinas donde se declaran y la única manera de enviar el o los valor(es) a otro subprograma es pasarlo(s) como parámetro(s). Sin embargo, esto puede ser inconveniente cuando el número de parámetros es muy grande. Para ello FORTRAN cuenta con lo que se conoce como un bloque común (common block), que se puede compartir con varios subprogramas.

Como ya se mencionó, un programa en FORTRAN puede contener varios subprogramas. Las variables que se declaren dentro de cada subprograma se conocen como variables locales (le pertenecen únicamente a la función o subrutina). Estas variables existen al momento de que el subprograma es llamado y desaparecen cuando éste llega a su fin.

```
integer function sumar()
    integer x
    !ámbito de la variable x desde aquí hasta el final de la
función
```

Las variables que se desean compartir entre varios subprogramas se llaman variables comunes. Las variables comunes existen durante la ejecución de todo el programa y pueden ser utilizadas por cualquier función o subrutina.



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	174/185
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 (Bloque común)

```
program commonBlock

    integer a, b
! Se comparten las variables a y b en un bloque común llamado global
    common /global/ a, b
    a = 5
    b = 4
    write(*,*) multiplicar()
    stop
    end

    integer function multiplicar()
        integer a, b
! Cualquier función puede ver el bloque común y puede utilizar los
valores
! o asignar valores a las variables del bloque.
        common /global/ a, b
        multiplicar = a*b
        return
    end function
```

En el ejemplo anterior, para comprobar que las variables que se están utilizando son, en efecto, las variables del bloque común, se puede comentar la invocación a este bloque dentro de la función multiplicar. Si no se utilizan las variables del bloque común entonces se utilizarán las variables a y b declaradas dentro de la función multiplicar, las cuales no han sido asignadas a valor alguno, dando como resultado cero en la operación multiplicación.

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	175/185
		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 (Bloque común)

```

program incrementaVariable

c La función incremento aumenta el valor de la
c variable enteraGlobal cada vez que es invocada.

c Solo la variable enteraGlobal es visible para
c otras funciones, ya que se encuentra en un bloque común

    integer enteraGlobal, enteraLocal
    common /global/ enteraGlobal

    do enteraLocal = 1,5,1
        write (*,*) incremento(), '+', enteraLocal, '='
        write (*,*) enteraGlobal + enteraLocal
    enddo

    stop
end

integer function incremento()
    integer enteraGlobal
    common /global/ enteraGlobal

    enteraGlobal = enteraGlobal + 2
    incremento = enteraGlobal

end function

```

### Bibliografía

- Oracle (2010). Fortran 77 Languaje 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/](http://web.stanford.edu/class/me200c/tutorial_77/)



**Manual de prácticas del  
Laboratorio de Programación  
básica**

Código:	MADO-18
Versión:	01
Página	176/185
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 13: Lectura y escritura de datos



**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

	<b>Manual de prácticas del Laboratorio de Programación básica</b>	Código:	MADO-18
		Versión:	01
		Página	177/185
		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 13: Lectura y escritura de datos

### **Objetivo:**

Elaborar programas en lenguaje FORTRAN que requieran el uso de archivos de texto plano en la resolución de problemas:

### **Actividades:**

- Abrir y cerrar un archivo.
- Leer y escribir en un archivo.

### **Introducción**

Un archivo es un conjunto de datos estructurados en una colección de entidades elementales o básicas denominadas registros que son del mismo tipo, pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Lenguaje FORTRAN permite manejar la entrada y la salida de datos desde o hacia un archivo, a través de funciones intrínsecas.

### **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.
 *
 * You should have received a copy of the GNU General Public License

```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	178/185
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

```
* along with this program. If not, see <http://www.gnu.org/licenses/>.  
*  
* Author: Jorge A. Solano  
*  
*/
```

## Número de unidad

En FORTRAN cada archivo está asociado a un número de unidad, que es un entero entre 1 y 99. Algunos números de unidad están reservados, por ejemplo: 5 para la entrada estándar (teclado) y 6 para la salida estándar (pantalla).

## Abrir archivo

La función `open()` se utiliza para abrir archivos, esto permite que FORTRAN pueda leer o escribir en él. Su estructura es la siguiente:

```
open(lista_de_especificadores)
```

Los especificadores más comunes son:

Especificador	Descripción
<code>[UNIT=] u</code>	Número de unidad. Un número entero entre 1 y 99 (Se puede elegir cualquier número pero éste debe ser único).
<code>IOSTAT= ios</code>	Identificador de estado de E/S. Debe ser una variable entera. Devuelve cero si la operación fue exitosa y cualquier otro número en caso de error.
<code>ERR= err</code>	Error. Etiqueta a la que el programa saltará si ocurre un error.
<code>FILE= fname</code>	Nombre del archivo. Cadena de caracteres que denota el nombre del archivo.
<code>STATUS= sta</code>	Modo de Apertura. Cadena de caracteres que puede tener los siguientes valores: <ul style="list-style-type: none"><li>• NEW: Crear un archivo nuevo.</li><li>• OLD: Abre un archivo existente.</li><li>• SCRATCH: Crea un archivo que existe solo durante la ejecución del programa y que se destruye con la instrucción de cerrar archivo (o cuando el programa termina).</li></ul>
<code>ACCESS= acc</code>	Tipo de Acceso. Cadena de caracteres que puede tener los siguientes valores:



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	179/185
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

- SEQUENTIAL
- DIRECT

Por defecto, el tipo de acceso es SEQUENTIAL.

**FORM=** *frm* Formato. Puede tomar los siguientes valores:

- FORMATTED
- UNFORMATTED

El formato por defecto es UNFORMATTED.

**RECL=** *rl* Especifica la longitud de cada registro en un archivo de acceso directo.

## Cerrar archivo

La función *close()* permite cerrar uno o varios archivos que fueron abiertos mediante una llamada a *open()*. La función *close()* permite escribir la información que se encuentre en el buffer hacia el disco y realiza un cierre formal del archivo a nivel del sistema operativo.

Un error en el cierre de un archivo puede generar todo tipo de problemas, incluyendo pérdida de datos, destrucción de archivos y posibles errores intermitentes en el programa. La estructura de esta función es:

```
close(Número_de_unidad[,lista_de_parametros])
```

Parámetro	Descripción
[UNIT=] <i>u</i>	Número de unidad, un entero único entre 1 y 99.
IOSTAT= <i>ios</i>	Identificador de estado de E/S. Debe ser una variable entera. Devuelve cero si la operación fue exitosa y cualquier otro número en caso de error.
ERR= <i>err</i>	Error. Etiqueta a la que el programa saltará si ocurre un error.
STATUS= <i>sta</i>	Modo de Cerrado. Cadena de caracteres que puede tener los valores: <ul style="list-style-type: none"><li>• KEEP: Valor por defecto.</li><li>• DELETE.</li></ul>



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	180/185
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

Los parámetros entre corchetes son opcionales. Si IOSTAT devuelve un valor cero significa que la operación de cierre ha tenido éxito. Generalmente, esta función solo falla cuando un disco se ha retirado antes de tiempo o cuando no queda espacio libre en el mismo.

Código (abrir y cerrar archivo existente)

```
program abrirCerrarArchivoExistente  
  
c Este programa permite abrir un archivo en modo lectura.  
  
integer error, u  
parameter(u=20)  
  
open(u, FILE='archivo.txt', STATUS='OLD', IOSTAT=error)  
  
if (error .EQ. 0) then  
    write (*,*) 'El archivo se abrió correctamente'  
    close(u)  
else  
    write (*,*) 'Error al abrir el archivo', error  
    write (*,*) 'El archivo no existe o no se'  
    write (*,*) 'tienen permisos de lectura.'  
endif  
  
stop  
end program
```

El programa anterior intenta abrir un archivo que ya existe, esto se especifica en el parámetro STATUS='OLD'. Si se desea abrir un archivo que no existe, es decir, crear un nuevo archivo, se debe especificar en el parámetro STATUS='NEW'.

Código (abrir y cerrar archivo existente)

```
program abrirCerrarArchivoNuevo  
  
c Este programa permite abrir un archivo en modo lectura.  
  
integer error, u  
parameter(u=20)  
  
open(u, FILE='archivo.txt', STATUS='NEW', IOSTAT=error)
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	181/185
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

```
if (error .EQ. 0) then
    write (*,*) 'El archivo se abrió correctamente'
    close(u)
else
    write (*,*) 'Error al abrir el archivo', error
    write (*,*) 'El archivo ya existe o no se'
    write (*,*) 'tienen permisos de escritura.'
endif

stop
end program
```

## Funciones READ y WRITE

Las funciones READ y WRITE permiten leer y escribir, respectivamente, datos sobre los archivos. Las estructuras de estas funciones son, respectivamente:

```
read([UNIT=] u, [FMT=] fmt, [IOSTAT=ios, ERR=err, END=s])
write([UNIT=] u, [FMT=] fmt, [IOSTAT=ios, ERR=err, END=s])
```

Donde el especificador END=s define a que etiqueta debe saltar el programa si se alcanza el fin del archivo. El especificador FMT permite utilizar los descriptores de formato.

La función *write()* permite escribir cualquier tipo de dato en un archivo específico. La función *read()* permite leer cualquier tipo de dato desde el archivo especificado. Esta función lee un renglón a la vez.

### Código (read)

```
program leerDeArchivo

c Este programa permite leer el contenido de un archivo.

integer error, u
character (len=5) caracteres
parameter(u=11)

open(u, FILE= 'archivo.txt', STATUS='OLD', IOSTAT=error)

if (error .EQ. 0) then
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	182/185
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 (*,*) 'El archivo se abrió correctamente'
write (*,*) 'El contenido del archivo es:'
do while (error .EQ. 0)
    read (u,*,IOSTAT=error) caracteres
    write (*,*) caracteres
enddo
close(u)
else
    write (*,*) 'Error al abrir el archivo', error
    write (*,*) 'El archivo no existe o no se'
    write (*,*) 'tienen permisos de lectura.'
endif

stop
end program
```

Tanto el número de unidad como el descriptor de formato son parámetros obligatorios para la función READ.

El número de unidad le indica a la función READ de donde va a obtener la información, en este caso se obtendrá del archivo abierto. El descriptor de formato permite imprimir la información en un formato personalizado, en este caso no está especificado y se indica con \*.

Debido a que la función READ lee una línea cada vez, la capacidad del arreglo donde se almacena la información (en este caso caracteres) debe ser del tamaño máximo de caracteres por renglón, en otro caso la información (el renglón) ser verá truncado.

Código (write)

```
program escribirEnArchivo
c Este programa permite escribir datos en un archivo.

integer error, u
character (len=35) escribir
parameter(u=15)
escribir = 'Escribir ésta cadena en archivo.'

open(u, FILE='escribirCadena.txt', STATUS='NEW', IOSTAT=error)

if (error .EQ. 0) then
    write (*,*) 'El archivo se creó correctamente'
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	183/185
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 (u,*) escribir
close (u)
else
    write (*,*) 'Error al crear el archivo', error
    write (*,*) 'El archivo ya existe o no se'
    write (*,*) 'tienen permisos de escritura.'
endif

stop
end program
```

### Código (Descriptores de formato)

```
program formato

c Este programa permite leer diferentes tipos de datos
c con un formato específico desde un archivo.

integer enteroUno, error, u
real realUno
complejo complejoUno
character caracterUno

parameter(u=9)

open(u, FILE='variables.txt', STATUS='OLD', IOSTAT=error)

if (error .EQ. 0) then
    read (u,100) enteroUno, realUno, caracterUno
100   format (I3,F6.3,A2)
    write (*,*) 'Entero: ', enteroUno
    write (*,*) 'Real: ', realUno
    write (*,*) 'Carácter: ', caracterUno
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	184/185
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

```
close (u)
endif
```

```
stop
end program
```

En el archivo de texto, las variables deben estar separadas por espacios para que puedan ser leídas de manera correcta por el programa, es decir:

Texto (variables.txt)

```
23 35.456 x
```



## Manual de prácticas del Laboratorio de Programación básica

Código:	MADO-18
Versión:	01
Página	185/185
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

## Bibliografía

- Oracle (2010). Fortran 77 Languaje 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/](http://web.stanford.edu/class/me200c/tutorial_77/)