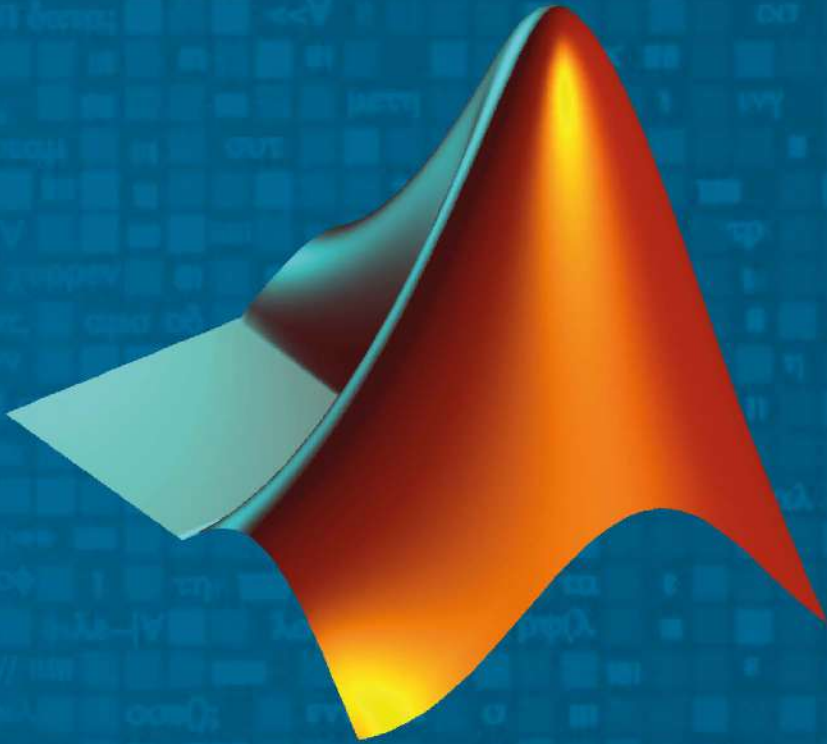


Programación con Matlab



*Alexandra O. Pazmiño A.
Jairo R. Jácome T.
Félix A. Ruiz M.*

CIDE
EDITORIAL



PROGRAMACIÓN CON MATLAB

PROGRAMACIÓN CON MATLAB

Alexandra O. Pazmiño A.

Escuela Superior Politécnica de Chimborazo

Jairo R. Jácome T.

Escuela Superior Politécnica de Chimborazo

Felix Ruiz.

Escuela Superior Politécnica de Chimborazo



Reservados todos los derechos. Está prohibido, bajo las sanciones penales y el resarcimiento civil previstos en las leyes, reproducir, registrar o transmitir esta publicación, íntegra o parcialmente, por cualquier sistema de recuperación y por cualquier medio, sea mecánico, electrónico, magnético, electroóptico, por fotocopia o por cualquiera otro, sin la autorización previa por escrito al Centro de Investigación y Desarrollo Ecuador (CIDE).

Copyright © 2023
Centro de Investigación y Desarrollo Ecuador
Tel.: + (593) 04 2037524
<http://www.cidecuador.com>

ISBN: 978-9942-616-27-2

DOI: <https://doi.org/10.33996/cide.ecuador.PM2616272>

Dirección editorial: Lic. Pedro Misacc Naranjo, Msc.

Coordinación técnica: Lic. María J. Delgado

Diseño gráfico: Lic. Danissa Colmenares

Diagramación: Lic. Alba Gil

Fecha de publicación: marzo, 2023



La presente obra fue evaluada por pares académicos
experimentados en el área.

Catalogación en la Fuente

Programación con MATLAB

Alexandra O. Pazmiño A.

Jairo R. Jácome T.

Félix A. Ruiz M.

Ecuador: Editorial CIDE, 2023

119 p.: incluye tablas, gráficos; 17,6 x 25,0 cm.

ISBN: 978-9942-616-27-2

DOI <https://doi.org/10.33996/cide.ecuador.PM2616272>

Semblanza de los autores

Alexandra Orfelina Pazmiño Armijos


Tecnóloga en Informática Aplicada
Ingeniera en Electrónica y Computación
Especialista en Redes de Comunicación de Datos
Magister en Informática Empresarial
Magister en Electrónica y Automatización
Docente Investigadora ESPOCH
Carrera de Ingeniería Industrial
apazmino_a@epoch.edu.ec

 <https://orcid.org/0000-0002-5111-7968>



Jairo René Jácome Tinoco


Tecnólogo en Informática Aplicada
Ingeniero en Electrónica y Computación
Magister en Sistemas de Telecomunicaciones
Técnico de Laboratorio Facultad de Mecánica ESPOCH
jjacome@epoch.edu.ec

 <https://orcid.org/0000-0002-3853-3107>



Félix Alejandro Ruiz Muñoz

Ingeniero Civil
Técnico de Laboratorio Facultad de Mecánica ESPOCH
felix.ruiz@epoch.edu.ec

 <https://orcid.org/0009-0007-4449-9025>



Prólogo

En la actualidad conocer de programación es de vital importancia para estudiantes y profesionales, para la lectura del libro es necesario conocer el funcionamiento básico del software Matlab el cual se utiliza para estudiar las diferentes estructuras que existen en la programación.

Es por ello que en el primer capítulo se hace un revisión rápida a Matlab recordando los principios básicos de su funcionamiento, cómo se ingresa y maneja vectores y matrices en el software, en el segundo capítulo se realiza una introducción a la Programación con ejercicios resueltos de estructuras secuenciales, estructuras de control y sus diferentes tipos como la Bifurcación simple, anidada, compuesta y múltiple.

Sin dejar de lado las estructuras de repetición se continúa con ejercicios resueltos con la estructura for y while. Finalmente en el capítulo se deja un número de ejercicios para resolver y practicar lo aprendido en el capítulo.

En el tercer capítulo es necesario abordar el tema de manejo de vectores y matrices utilizando estructuras de control y repetición, de tal manera que se permita manejar la información que esta almacenada, accediendo a cada una de las posiciones de estas variables estructuradas.

En el cuarto y último capítulo se trata el tema de funciones definidas por el usuario, al Matlab ser un software modular, se resuelve ejercicios utilizando lo aprendido en los capítulos anteriores, creando los diferentes tipos de funciones como funciones con una sola entrada y una sola salida, funciones con varias entradas y una salida, funciones con varias entradas y varias salidas, permitiendo crear programas en módulos mucho más fácil de leer y dar solución a ejercicios planteados.

En cada uno de los capítulos se ha dejado ejercicios por resolver de tal manera que el lector pueda practicar lo aprendido y reforzar sus conocimientos.

Índice general

1. INTRODUCCIÓN A MATLAB	14
1.1. Qué es Matlab	14
1.2. Ventajas de Matlab	14
1.3. Entorno de Matlab	15
1.3.1. Command Window o Ventana de comandos	15
1.3.2. Workspace o Espacio de trabajo	16
1.3.3. Current folder o Carpeta actual	16
1.3.4. Editor	17
1.3.5. Command History	17
1.4. Operadores de Matlab	18
1.4.1. Operadores Aritméticos	18
1.4.2. Operadores de relación	20
1.4.3. Operadores Lógicos	21
1.5. Limpiar la ventana de comandos y el Workspace	22
1.6. Ejercicios propuestos	22
1.7. Despliegue de números	22
1.7.1. Notación científica:	22
1.7.2. Formato de despliegue	22
1.8. Introducción a vectores y matrices	24
1.8.1. Definición de vectores o arreglos unidimensionales	24
1.8.2. Generación rápida de vectores	24
1.8.3. Operaciones con arreglos	26
1.8.4. Funciones elementales con un vector complejo como argumento	29
1.8.5. Definición de matrices o arreglos bidimensionales	31
1.8.6. Funciones para crear matrices	32
1.8.7. Creación y concatenación de matrices	33
1.8.8. Uso del operador dos puntos	34
1.8.9. Características de la matriz	37
1.8.10. Matrices especiales	37
1.8.11. Funciones	39
1.9. Ejercicios propuestos	41
2. INTRODUCCIÓN A LA PROGRAMACIÓN EN MATLAB	43
2.1. Tipos de archivos de Matlab	43
2.1.1. Script	43
2.1.2. Function	43
2.2. Entradas y salidas controladas por el usuario	43
2.2.1. Función input	44

2.2.2.	Opciones de Salida	44
2.3.	Tipos de Estructuras	46
2.4.	Estructuras secuenciales	47
2.4.1.	Ejercicios propuestos estructuras secuenciales	50
2.5.	Estructuras de Control	50
2.5.1.	Bifurcación simple	51
2.5.2.	Bifurcación anidada	54
2.5.3.	Bifurcación compuesta	59
2.5.4.	Bifurcación Múltiple	60
2.5.5.	Ejercicios propuestos estructuras de control	63
2.6.	Estructuras de repetición	64
2.6.1.	Estructura while - do	64
2.6.2.	Estructura for	69
2.6.3.	Sentencia break	75
2.6.4.	Sentencia continue	76
2.6.5.	Ejercicios Propuestos	77
3.	MANEJO DE VECTORES Y MATRICES CON ESTRUCTURAS	80
3.1.	Arreglos unidimensionales o vectores	80
3.1.1.	Lectura de un vector	80
3.1.2.	Escritura de un vector	82
3.1.3.	Operaciones con vectores	82
3.1.4.	Ejercicios propuestos	86
3.2.	Arreglos bidimensionales o vectores	86
3.2.1.	Lectura de un matriz	86
3.2.2.	Escritura de una matriz	87
3.2.3.	Operaciones con matrices	87
3.2.4.	Ejercicios propuestos	92
4.	FUNCIONES DEFINIDAS POR EL USUARIO	95
4.1.	Declaración de funciones	95
4.2.	Tipos de funciones	96
4.2.1.	Funciones con una sola entrada y una sola salida	97
4.2.2.	Funciones con varias entradas y una sola salida	100
4.2.3.	Funciones con una sola entrada y varias salidas	101
4.2.4.	Funciones con varias entradas y varias salidas	102
4.3.	Ejercicios resueltos de funciones	102
4.4.	Ejercicios propuestos	107
5.	SOLUCIÓN EJERCICIOS PROPUESTOS	110
5.1.	Solución Ejercicios propuestos Capitulo 1 numeral 1.6	110
5.2.	Solución Ejercicios propuestos capitulo 1 numeral 1.9	111
5.3.	Solución Ejercicios propuestos Capitulo 2	111
5.4.	Solución Ejercicios propuestos Capítulo 3	114
5.5.	Solución Ejercicios propuestos Capítulo 4	116

Índice de figuras

1.1. Logo de Matlab	14
1.2. Ventana de comandos	15
1.3. Operador ;	15
1.4. Espacio de trabajo	16
1.5. Carpeta actual	17
1.6. Detalles de la carpeta actual	17
1.7. Ventana del Editor	18
1.8. Historial de comandos	18
2.1. Nuevo script	43
2.2. Archivo de una función	44
2.3. Formateo de la salida fprintf	45
2.4. Salida ejemplo 1 estructuras secuenciales	47
2.5. Salida ejemplo 2 estructuras secuenciales	47
2.6. Salida ejemplo 3 estructuras secuenciales	47
2.7. Salida ejemplo 4 estructuras secuenciales	48
2.8. Salida ejemplo 5 estructuras secuenciales	48
2.9. Salida ejemplo 6 estructuras secuenciales	48
2.10. Salida ejemplo 7 estructuras secuenciales	49
2.11. Salida ejemplo 8 estructuras secuenciales	49
2.12. Salida ejemplo 9 estructuras secuenciales	49
2.13. Salida ejemplo 10 estructuras secuenciales	50
2.14. Salida ejemplo 1 estructuras de bifurcación	51
2.15. Salida ejemplo 2 estructuras de bifurcación	51
2.16. Salida ejemplo 3 estructuras de bifurcación	52
2.17. Salida ejemplo 4 estructuras de bifurcación	52
2.18. Salida ejemplo 5 estructuras de bifurcación	53
2.19. Salida ejemplo 6 estructuras de bifurcación	53
2.20. Salida ejemplo 7 estructuras de bifurcación	54
2.21. Salida ejemplo 8 estructuras de bifurcación	54
2.22. Salida ejemplo 9 estructuras de bifurcación	55
2.23. Salida ejemplo 10 estructuras de bifurcación	65
2.24. Salida ejemplo 11 estructuras de bifurcación	57
2.25. Salida ejemplo 12 estructuras de bifurcación	58
2.26. Salida ejemplo 13 estructuras de bifurcación	59
2.27. Salida ejemplo 14 estructuras de bifurcación	59
2.28. Salida ejemplo 15 estructuras de bifurcación	60
2.29. Salida ejemplo 16 estructuras de bifurcación	60
2.30. Salida ejemplo 17 estructuras de bifurcación	61
2.31. Salida ejemplo 18 estructuras de bifurcación	62
2.32. Salida ejemplo 19 estructuras de bifurcación	62
2.33. Salida ejemplo 20 estructuras de bifurcación	63

2.34. Salida ejemplo 1 estructuras de repetición	64
2.35. Salida ejemplo 2 estructuras de repetición	65
2.36. Salida ejemplo 3 estructuras de repetición	65
2.37. Salida ejemplo 4 estructuras de repetición	66
2.38. Salida ejemplo 5 estructuras de repetición	66
2.39. Salida ejemplo 6 estructuras de repetición	66
2.40. Salida ejemplo 7 estructuras de repetición	67
2.41. Salida ejemplo 8 estructuras de repetición	67
2.42. Salida ejemplo 9 estructuras de repetición	68
2.43. Salida ejemplo 10 estructuras de repetición	68
2.44. Salida ejemplo 11 estructuras de repetición	70
2.45. Salida ejemplo 12 estructuras de repetición	70
2.46. Salida ejemplo 13 estructuras de repetición	71
2.47. Salida ejemplo 14 estructuras de repetición	71
2.48. Salida ejemplo 15 estructuras de repetición	72
2.49. Salida ejemplo 16 estructuras de repetición	73
2.50. Salida ejemplo 17 estructuras de repetición	73
2.51. Salida ejemplo 18 estructuras de repetición	74
2.52. Salida ejemplo 19 estructuras de repetición	75
2.53. Salida ejemplo 20 estructuras de repetición	75
2.54. Salida ejemplo instrucción break	76
2.55. Salida ejemplo sentencia continue	76
3.1. Salida ejemplo 1 vectores	81
3.2. Salida ejemplo 2 vectores	81
3.3. Salida ejemplo 3 vectores	82
3.4. Salida ejemplo 4 vectores	82
3.5. Salida ejemplo 5 vectores	83
3.6. Salida ejemplo 6 vectores	83
3.7. Salida ejemplo 7 vectores	84
3.8. Salida ejemplo 8 vectores	84
3.9. Salida ejemplo 9 vectores	85
3.10. Salida ejemplo 10 vectores	85
3.11. Salida ejemplo 1 matrices	87
3.12. Salida ejemplo 2 matrices	87
3.13. Salida ejemplo 3 matrices	88
3.14. Salida ejemplo 4 matrices	88
3.15. Salida ejemplo 5 matrices	89
3.16. Salida ejemplo 6 matrices	90
3.17. Salida ejemplo 7 matrices	90
3.18. Salida ejemplo 8 matrices	91
3.19. Salida ejemplo 9 matrices	92
3.20. Salida ejemplo 10 matrices	92
4.1. Crear archivo para una función	96
4.2. Plantilla de una función	96
4.3. Ejercicio de funciones	100

Índice de tablas

1.1. Ventajas	14
1.2. Operadores Aritméticos	18
1.3. Jerarquía operadores aritméticos	19
1.4. Operadores Relación	20
1.5. Operadores Lógicos	21
2.1. Caracteres de conversión para formatear datos numéricos y de caracteres	45
2.2. Comandos de formato especial	46
2.3. Tipos de condiciones	50

Introducción

MATLAB es el nombre abreviado de “MATrix LABoratory”. MATLAB es un software para realizar cálculo numérico con datos escalares reales y complejos, vectores y matrices. El mayor potencial del software es su capacidad de procesar los datos con matrices para almacenar datos, realizar gráficos y realizar programas o scripts con un lenguaje de programación propio.

Aprender la forma correcta de definir vectores o matrices, así como las diferentes estructuras de control condicionales y repetición permiten resolver problemas a través de la programación, los ejercicios resueltos ayudan a desarrollar y comprender de mejor manera las diferentes estructuras.

Se plantea un libro como una guía básica para favorecer los diferentes ritmos de aprendizaje con ejercicios que se basan en los temas desarrollados, mostrando una solución fácil de entender y que permita al lector tener una idea clara de como se puede resolver el planteamiento y la posibilidad que pueda abordarlo desde otras perspectivas si así fuese necesario, con el uso de las diferentes tipos de estructuras estudiadas.

Matlab es un programa que se caracteriza por trabajar con funciones, esto permite tener un código mucho mas fácil de leer e interpretar así como también se evita repetir varias veces la misma programación haciendo uso de los diferentes tipos de funciones que se pueden crear.

Revisar conceptos básicos del manejo de Matlab, de programación y probar cada uno de los ejercicios planteados en el software permiten tener una mejor comprensión, los ejercicios propuestos permiten comprobar lo aprendido en el estudio de cada uno de los capítulos, sin embargo se ha incluido un capítulo con la solución para su verificación.

1

Introducción a Matlab



1.1. Qué es Matlab

MATLAB (abreviatura de **MA**TriX **LAB**oratory, «laboratorio de matrices») es un sistema de cómputo numérico que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, macOS y GNU/Linux.

MATLAB no siempre es la mejor herramienta para usar en una tarea de programación. El programa destaca en cálculos numéricos, especialmente en los relacionados con matrices y gráficas, puesto que MATLAB es óptimo para matrices, si un problema se puede formular con una solución matricial, MATLAB lo ejecuta sustancialmente más rápido que un programa similar en un lenguaje de alto nivel.



Figura 1.1: Logo de Matlab

1.2. Ventajas de Matlab

Tabla 1.1: Ventajas

Ventajas de usar Matlab	Desventajas de usar Matlab
Amplio soporte de funciones ya desarrolladas	Gestión “oscura” de la memoria
Rápido prototipado	Problemas eventuales de velocidad
Integración con dispositivos hardware	Distribución de ejecutables
Toolboxes para aplicación en ingeniería e investigación	Es preciso linkar con librerías numéricas y gráficas.

1.3. Entorno de Matlab

1.3.1. Command Window o Ventana de comandos

Se encuentra ubicado en la parte central de Matlab, ésta es la ventana más importante, ya que en ella se deben teclear las instrucciones a ejecutar, apareciendo el resultado de inmediato.



Figura 1.2: Ventana de comandos

Es el principal mecanismo para trabajar con MATLAB. Las funciones introducidas, llamadas también entradas se ejecutan pulsando la tecla Enter. Se debe tener en cuenta que al escribir los nombres de las funciones o de los comandos, MATLAB distingue entre mayúsculas y minúsculas (por lo general, las funciones se escriben en minúsculas).

A continuación una lista de comandos útiles en la línea de comandos:

clc: Borra el contenido de la pantalla y coloca el cursor en la primera línea.

Esc: Borra la línea.

%: Todo lo que aparece detrás del símbolo % y en la misma línea se considera un comentario.

ctrl+c: Detiene la ejecución de cualquier comando o función.

A veces también es muy importante, que el resultado de un cálculo no aparezca en pantalla. Por ejemplo, si generamos una matriz de orden muy alto con el objeto de hacer después una gráfica, todos los elementos se visualizarán en la pantalla y esta se llena completamente. Para evitar mostrarlo en la pantalla ponemos un punto y coma (;) al final de la instrucción.

```
>> r=cos(pi)
r =
    -1
>> r=cos(pi);
>> |
```

Muestra el valor

No muestra el valor

Figura 1.3: Operador ;

Los comandos se pueden ejecutar o escribir uno a uno, y también se puede escribir uno a continuación de otro en una misma línea, para lo cual deben ir separados por comas (,).

```
>> x=0,y=0,z=0
x =
    0
y =
    0
z =
    0
```

Si el comando o la cantidad de comandos son demasiado larga para que aparezca en un único renglón, se puede romper la cadena y seguir en el siguiente renglón, escribiendo tres puntos suspensivos (...+[ENTER]).

1.3.2. Workspace o Espacio de trabajo

En la esquina inferior izquierda aparece la ventana de espacio de trabajo (Workspace) que guarda la información de las variables utilizadas en la sesión de trabajo actual. Para cada variable se muestra su nombre y tamaño.

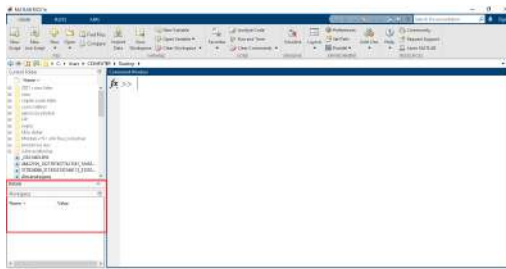


Figura 1.4: Espacio de trabajo

Para ver información acerca de las variables que se esta utilizando en Matlab se puede utilizar los comandos:

who para obtener la lista de las variables (no de sus valores).

whos para obtener la lista de las variables e información del tamaño, tipo y atributos (tampoco da valores).

```
>> who
Your variables are:
a      b      c      d
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
a	1x1	8	double	
b	1x3	24	double	
c	2x3	48	double	
d	1x1	8	double	

1.3.3. Current folder o Carpeta actual

La ventana folder actual se sitúa en la parte izquierda del escritorio de MATLAB. La carpeta actual es una ubicación de referencia que MATLAB utiliza para buscar archivos. Esta carpeta a veces se denomina directorio actual, carpeta de trabajo actual o directorio de trabajo actual.

Esta ventana se divide en 2 partes, Name y details. En la primera se muestra los ficheros del entorno de MATLAB que se puede abrir o hacer cambios y en la segunda se muestra los detalles que pueden tener estos ficheros como por ejemplo si son funcion o script además de los comentarios que puedan presentar.



Figura 1.5: Carpeta actual

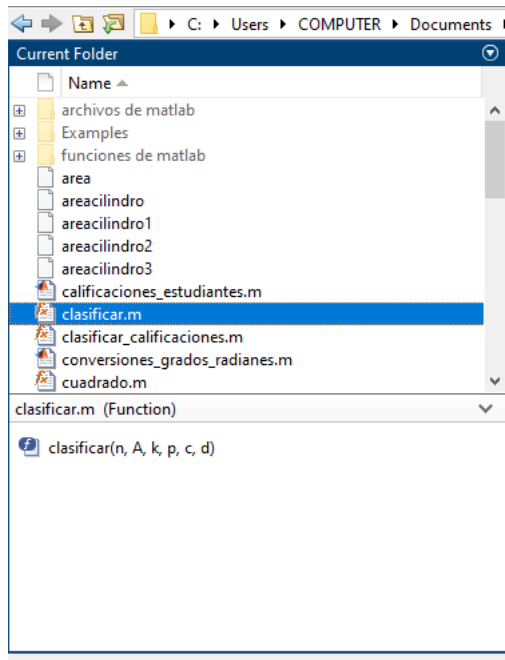


Figura 1.6: Detalles de la carpeta actual

1.3.4. Editor

MATLAB Editor. Cree scripts con una combinación de código, salida y texto formateado en un cuaderno ejecutable. La ventana se muestra en la Figura 1.7

1.3.5. Command History

En esta ventana de la Figura 1.8 se muestra un registro de las instrucciones que ejecutó en las sesiones actuales y anteriores. El historial de comandos enumera la hora y la fecha de cada sesión en el formato de fecha corta para su sistema operativo, seguido de las declaraciones de esa sesión. Esta ventana puede estar anclada a Matlab o no, se puede recuperar los comandos ejecutados en sesiones anteriores o actuales utilizando las flechas en el teclado hacia arriba o hacia abajo.

Con las flechas del cursor ($\uparrow\downarrow$) se pueden recuperar las órdenes anteriores, sin tener que volver a escribirlas. La flechas ($\leftarrow\rightarrow$) permiten presentar el desplazamiento horizontal en la línea de comandos. Estas flechas son de mucha utilidad en el caso de una equivocación o cuando queremos volver a ejecutar un mismo comando o hacerle una pequeña modificación.



Figura 1.7: Ventana del Editor

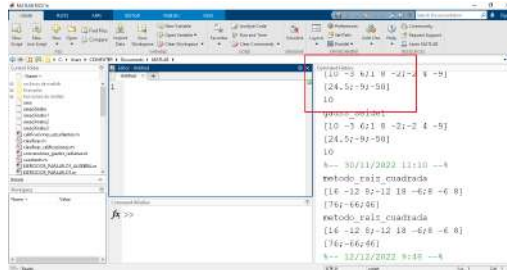


Figura 1.8: Historial de comandos

1.4. Operadores de Matlab

Matlab utiliza varios operadores dentro de los cuales podemos mencionar:

- Operadores Aritméticos
- Operadores Lógicos
- Operadores de Relación

1.4.1. Operadores Aritméticos

Son los que se utilizan para realizar operaciones entre operandos.

Tabla 1.2: Operadores Aritméticos

Operador	Operación
+	Suma
-	Resta
*	Multiplicación
/, \	División izquierda, División derecha
^	Potencia
sqrt	Raíz cuadrada
floor	Toma el cociente de una división
mod, rem	Toma el residuo de una división

Ejemplos:

```
>> 5+6
ans =
    11
>> 7-2
ans =
     5
```

```

>> 3*56
ans =
    168
>> 1/2
ans =
    0.5000
>> 1\2
ans =
     2
>> 3^5
ans =
    243
>> sqrt(36)
ans =
     6
>> floor(9/2)
ans =
     4
>> mod(9,2)
ans =
     1
>> rem(9,2)
ans =
     1

```

Jerarquía de las Operaciones

Cuando se trabaja con operadores aritméticos es necesario conocer el orden en el que Matlab resuelve las operaciones, cuando hay operadores con la misma jerarquía resuelve de izquierda a derecha.

1. Destrucción de signos de agrupación
2. Potencia y Raíz
3. Multiplicación y división
4. Suma y resta
5. Cambio de signo

Tabla 1.3: Jerarquía operadores aritméticos

Nombre del Operador	Operador
Paréntesis	(), [], { }
Potencia y raíz	^ ; √
Multiplicación y división	*, /
Suma y resta	+, -
Cambio de signo	(-)

Ejemplos:

Realizar los siguientes cálculos

- -1^4
ans =
-1
- $(-1)^4$
ans =
1
- $2/3^2$
ans =
0.2222
- $6^2 - 4 - \frac{7}{2} * 3$
ans =
21.5000
- $6^2 - 4 - \frac{7}{2*3}$
ans =
30.8333
- $5 * 4^2 + 7$
ans =
87
- $(5 * 4)^2 + 7$
ans =
407

1.4.2. Operadores de relación

Permiten realizar comparaciones de dos operandos que pueden ser números, caracteres, cadena de caracteres, constantes o variables. Estos operadores sirven para expresar las condiciones. Devuelve Matlab un 1 lógico si es verdadero y 0 lógico si es falso.

Tabla 1.4: Operadores Relación

Operador	Significado
<, <=	menor, menor igual
>, >=	mayor, mayor igual
==	igual
~=	diferente

Ejemplos:

- `>> 5 > 3`
ans =
logical
1
- `>> 6 <= 2`
ans =
logical
0
- `>> 7 >= 5`
ans =
logical
1

- `>> 9 <= 12`
`ans =`
`logical`
`1`
- `>> 9 == 9`
`ans =`
`logical`
`1`
- `>> 9 = 9`
`ans =`
`logical`
`0`

1.4.3. Operadores Lógicos

Son utilizados para realizar comparaciones entre dos valores booleanos es decir devuelve un resultado verdadero o falso.

Tabla 1.5: Operadores Lógicos

Operador	Relación	Concepto
<code>&</code>	AND o conjunción	Es verdadera si todas las preguntas son verdaderas, si una es falsa se va por la opción falso.
<code> </code>	OR o disyunción	Es verdadera si por lo menos una de las preguntas es verdadera, y es la condición falsa cuando todas las condiciones son falsas.

Ejemplos:

- `>> 5 < 3&4 > 3`
`ans =`
`logical`
`0`
- `>> 5 >= 9|9 > 3`
`ans =`
`logical`
`1`
- `>> 6 == 6&6 < 2`
`ans =`
`logical`
`0`
- `>> 8 = 9|8 == 5`
`ans =`
`logical`
`1`

1.5. Limpiar la ventana de comandos y el Workspace

Para eliminar todo el texto escrito en la ventana de comandos se utiliza el comando *clc*. La ejecución de esta orden no afecta a las variables de la sesión de trabajo (la ventana Workspace sigue manteniendo las variables)

Para vaciar el Workspace es decir el contenido de las variables utilizamos el comando *clear*, y si deseamos borrar una sola variables utilizamos *clear nombrevariable*

1.6. Ejercicios propuestos

Escriba las siguientes expresiones en el prompt de Matlab utilizando correctamente los operadores aritméticos y signos de agrupación.

1. $1 + \frac{3}{4}$
2. $\frac{5+3}{9-1}$
3. $2^3 - \frac{4}{5+3}$
4. $4\frac{1}{2} * 5\frac{2}{3}$
5. $9\frac{6}{12} + 7 * 5^{3+2}$
6. $2 * (5 - \frac{3^2}{4}) + (-2^{-3})$
7. $\frac{5-(1/2)^2}{0,7+1}$
8. $(1 - 0,25)^{1/2} + (4/81)^{-1/2}$
9. $\sqrt{\sqrt{256} + \sqrt{(1-25)^{-2}}}$
10. $\frac{(5-i)^2}{3i} + \sqrt{2-i}$

1.7. Despliegue de números

1.7.1. Notación científica:

La Notación científica expresa un valor como un número entre 1 y 10 multiplicado por una potencia de 10. En Matlab se designan con una e entre el numero decimal y el exponente.

```
>> a = 9,00452e13
```

No debe existir espacios en blanco entre el numero decimal y el exponente.

```
>> 9,00452e13
```

1.7.2. Formato de despliegue

Matlab usa en sus cálculos números punto flotante , de cuantos dígitos se usen depende de su cálculo. Los enteros se imprimen sin punto decimal, los valores con fracciones decimales se imprimen en el formato corto por defecto muestra 4 dígitos decimales.

Ejemplos:

- ```
>> 6,5
ans =
6.5000
```



```
■ >> 7
 ans =
 7
```

MATLAB permite especificar otros formatos que muestren dígitos significativos adicionales.

- **format long (formato largo)**: Despliega en un formato decimal de 14 dígitos decimales.
- **format bank (formato banco)**: se despliega dos dígitos decimales.
- **format short (formato corto)**: Regresa el formato a 4 dígitos decimales.
- **format short e (formato corto con notación científica)**: despliega los números en notación científica con cuatro dígitos decimales.
- **format long e (formato largo con notación científica)**: despliega los números en notación científica con 14 dígitos decimales.
- **format + (formato mas)**: los únicos caracteres que se imprimen son los signos mas y menos.
- **format rat (formato fracciones)**: despliega números como números racionales (fracciones)

*Ejemplos:*

```
>> format long
>> 9.76678
ans =
 9.766780000000001

>> format short
>> 9.5678954
ans =
 9.5679

>> format bank
>> 33.4587
ans =
 33.46

>> format short e
>> 0.03838329
ans =
 3.8383e-02

>> format long e
>> 0.03838329
ans =
 3.838329000000000e-02

>> format +
>> -5
ans =
-

>> format rat
>> 0.5
ans =
 1/2
```

## 1.8. Introducción a vectores y matrices

### 1.8.1. Definición de vectores o arreglos unidimensionales

Para ingresar un vector en Matlab no hace falta conocer su tamaño, simplemente, se definen los valores de los elementos que van a componer el vector entre corchetes, separados por espacios o una coma, en el caso de vectores fila, o por el carácter punto y coma (;) en el caso de vectores columna.

**Ejemplo:**

```
>> b = [2 4 6 8 10] (separados los elementos por espacios)
```

o bien

```
>> b = [2,4,6,8,10] (separados los elementos por comas)
```

se genera el vector fila

```
b=
 2 4 6 8 10
```

Mientras que:

```
>> c = [6;2;9;4]
```

se genera el vector columna

```
c=
 6
 2
 9
 4
```

Para acceder a los elementos de un vector se utiliza un número entero llamado índices. Los índices correspondientes a los elementos de un vector comienzan en uno.

**Ejemplo:**

```
>>c(3)
```

```
ans = 9
```

Debido a que Matlab trabaja todo en base a matrices, se denominan vectores por tener una sola fila o una sola columna sin embargo es una matriz y su tamaño está definido por número de filas y número de columnas. Por lo tanto podemos acceder a un elemento del mismo con dos índices que corresponden al número de fila y número de columna que corresponde.

**Ejemplo:**

```
>>c(3,2)
```

```
ans = 9
```

### 1.8.2. Generación rápida de vectores

#### 1. Operador (:)

No necesariamente se debe escribir de forma explícita todos sus elementos existen otras formas de generación de vectores, para ello se utiliza el operador dos puntos(:) de la siguiente forma

```
variable=[vin:vfin]
```

Esta declaración define un vector con elementos que se forman con `vin` como el primer elemento y el último elemento por `vfin`, los componentes intermedios están separados por una unidad. Se puede utilizar corchetes, paréntesis o ninguno al realizar la declaración del vector.

**Ejemplo:**

```
>> s = 1 : 10
```

```
s = 1 2 3 4 5 6 7 8 9 10
```

```
variable=[vin:incr:vfin]
```

Define un vector cuyo primer elemento está especificado por `vin` y el último elemento por un valor menor o igual a `vfin`, los componentes intermedios están separados por un valor definido en `incr`. De la misma forma para la declaración se puede utilizar corchetes, paréntesis o ninguno.

**Ejemplos:**

```
>>c=3:2:10
```

```
c =
 3 5 7 9
```

```
>> v = 0 : pi/2 : 2 * pi
```

```
v =
 0 1.57 3.14 4.71 6.28
```

```
>> p = 20 : -5 : 0
```

```
p =
 20.00 15.00 10.00 5.00 0
```

## 2. Función `linspace`

Esta función de matlab genera un vector con `n` valores igualmente espaciados entre `x1` y `x2`.

```
variable=linspace(x1,x2)
```

Genera un vector fila de 100 elementos equidistantes entre `x1` y `x2`.

```
variable=linspace(x1,x2,n)
```

Genera un vector fila de `n` elementos equidistantes entre `x1` y `x2`.

**Ejemplos:**

```
>> v = linspace(5, 20, 5)
```

```
v =
 5.00 8.75 12.50 16.25 20.00
```

```
>> v = linspace(5, 20, 8)
```

```
v =
 5.00 7.14 9.29 11.43 13.57 15.71 17.86 20.00
```

El espaciado entre un elemento y otro lo determina matlab para generar la cantidad de `n` elementos que se desea en el vector.

3. Función logspace Generar un vector espaciado logarítmicamente.

```
variable=logspace(x1,x2)
```

Genera un vector fila de 50 elementos espaciados logarítmicamente entre  $10^{x1}$  y  $10^{x2}$

```
variable=logspace(x1,x2,n)
```

Genera n elementos entre  $10^{x1}$  y  $10^{x2}$

```
variable=logspace(x1,pi)
```

Genera un vector fila de 50 elementos entre  $10^{x1}$  y pi

```
variable=logspace(x1,pi,n)
```

Genera n elementos entre  $10^{x1}$  y pi

**Ejemplos:**

```
>> s = logspace(1,5,7)
```

```
s =
```

```
 1.0e+05 *
 0.0001 0.0005 0.0022 0.0100 0.0464 0.2154 1.0000
```

Genere un vector s de 7 elementos espaciados logarítmicamente en el intervalo  $[10^1, 10^5]$ .

### 1.8.3. Operaciones con arreglos

Una vez se definen los vectores se puede realizar operaciones con ellos sin olvidar que para Matlab es una matriz.

Podemos realizar operaciones de suma, resta, multiplicación y división de un vector con un valor escalar.

**Ejemplos:**

```
>> a=[3 6 7 1 9]
```

```
a =
```

```
 3.00 6.00 7.00 1.00 9.00
```

```
>> b=a+3
```

```
b =
```

```
 6.00 9.00 10.00 4.00 12.00
```

```
>> c = 9 - a
```

```
c =
```

```
 6.00 3.00 2.00 8.00 0
```

```
>> d = a * 5
```

```
d =
```

```
 15.00 30.00 35.00 5.00 45.00
```

```
>> s = a/2
```

```
s =
```

```
 1.50 3.00 3.50 0.50 4.50
```

Para realizar operaciones entre vectores hay que tomar en cuenta algunas consideraciones como por ejemplo para la suma y resta entre vectores estos deben tener la misma dimensión.

**Ejemplos:**

```
>> a=[4 6 2 5];
>> b=[5 4 6 1];
>> c=a+b
c =
 9.00 10.00 8.00 6.00
```

```
>> a=[4 6 2 5]
>> b=[5 4 6 1];
>> d=a-b
d =
-1.00 2.00 -4.00 4.00
```

Para lo que es multiplicación de matrices se debe considerar las dimensiones de las matrices, la primera debe tener el mismo número de columnas que filas de la segunda. La dimensión de la matriz resultante del producto tendrá el mismo número de filas de la primera con el mismo número de columnas de la segunda.

Por lo tanto si ingresamos en matlab dos vectores de la siguiente manera y realizamos un producto:

```
>> a=[4 6 2 5];
>> b=[5 4 6 1];
>> c=a*b
```

Error using \* Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in the second matrix. To perform elementwise multiplication, use .\*.

Related documentation

Nos arroja ese error debido a que no cumple con las condiciones mencionadas anteriormente para realizar el producto. Sin embargo se puede realizar la multiplicación de los vectores elemento a elemento utilizando el operador punto(.) antes del operador aritmético.

```
>> c=a.*b
c =
 20.00 24.00 12.00 5.00
```

Lo mismo se debe considerar al realizar una operación de división, potencia si lo que se desea es hacerlo elemento a elemento.

### ***Ejemplos:***

```
>> a=[4 6 2 5];
>> b=[5 4 6 1];
>> c=a./b
```

```
c =
 0.80 1.50 0.33 5.00
```

```
>> d=a.^2
d =
 16.00 36.00 4.00 25.00
```

Si a es un vector fila y b un vector columna, con la misma cantidad de elementos, entonces la operación a\*b' resulta un número, es decir representa el producto escalar (o producto interno) de a y b.

```
>> a=1:5, b=5:-1:1
a =
 1 2 3 4 5
```

```
b =
 5 4 3 2 1
```

```
>> a*b'
ans =
 35
```

**Ejemplos:**

1. Crear un vector que va del 2 al 12 con incremento de 2, y mostrar los elementos  $y(6)$ ,  $y(3)$ ,  $y(11)$  en un vector en el orden mencionado.

```
>> y=2:15
y =
 2 3 4 5 6 7 8 9 10 11 12

>> z=[y(6) y(3) y(11)]
z =
 7 4 12
```

2. Crear un vector  $s$  que va de 1 a 21 con incremento de 5.  
Crear otro vector  $t$  con los mismos elementos de  $s$ , de forma que la suma  $s+t$  sea un vector con todos sus elementos iguales.

```
>> s=1:5:21
s =
 1 6 11 16 21

>> t=21:-5:1
t =
 21 16 11 6 1

>> r=s+t
r =
 22 22 22 22 22
```

3. Crear un vector  $s$  compuesto por 20 elementos igualmente espaciados en  $[0, 2\pi]$ .  
Determine los vectores  $s_1$  y  $s_2$  que representan las funciones seno y tangente de los valores del vector  $s$ .  
Subdividir los elementos de  $s_1$  en un vector  $t_1$  que contenga las posiciones impares y un vector  $t_2$  con las posiciones pares.  
Mostrar un vector  $t_3$  que contenga el cuadrado de los elementos de la suma de  $s_1$  y  $s_2$ .

```
>> s=linspace(0,2*pi,20);
>> s=linspace(0,2*pi,20);
>> s1=sin(s);
>> s2=tan(s);
```

```
>> t1=s(1:2:20);
>> t2=s(2:2:20);
>> t3=(s1+s2).^2;
```

#### 1.8.4. Funciones elementales con un vector complejo como argumento

MATLAB trabaja además con funciones de variable compleja en este caso se utiliza S como argumento de la función, esta variable puede ser vectorial o matricial. Algunas de las funciones que también trabaja con variable compleja vectorial que tiene MATLAB se muestran a continuación

- **max(S)**: Mayor componente (para complejos se calcula  $\max(\text{abs}(V))$ ).
- **min(S)**: Menor componente (para complejos se calcula  $\min(\text{abs}(V))$ ).
- **mean(S)**: Media de las componentes de V.
- **median(S)**: Mediana de las componentes de V.
- **std(S)**: Desviación típica de las componentes de V.
- **sort(S)**: Ordena de forma ascendente las componentes de V. Para complejos hace la ordenación según los valores absolutos.
- **sum(S)**: Suma las componentes de V.
- **prod(S)**: Multiplica los elementos de V, con lo que  $n! = \text{prod}(1:n)$
- **cumsum(S)**: Da el vector de sumas acumuladas de V.
- **cumprod(S)**: Da el vector de productos acumulados de V.

#### *Ejemplos:*

```
>> a=[6 4 2 11 9 7], b=[9+2i,-1+3i,5i,-6-5i,4]
a =
 6 4 2 11 9 7

b =
 9.0000 + 2.0000i -1.0000 + 3.0000i 0.0000 + 5.0000i -6.0000 - 5.0000i 4.0000
+ 0.0000i

>> max(a),max(b)
ans =
 11

ans =
 9.0000 + 2.0000i

>> min(a),min(b)
```

```

ans =
 2

ans =
 -1.0000 + 3.0000i

>> mean(a),mean(b)
ans =
 6.5000

ans =
 1.2000 + 1.0000i

>> median(a),median(b)
ans =
 6.5000

ans =
 0.0000 + 5.0000i

>> sort(a),sort(b)
ans =
 2 4 6 7 9 11

ans =
 -1.0000 + 3.0000i 4.0000 + 0.0000i 0.0000 + 5.0000i -6.0000 - 5.0000i 9.0000
+ 2.0000i

>> sum(a),sum(b)
ans =
 39

ans =
 6.0000 + 5.0000i

>> prod(a),prod(b)
ans =
 33264

ans =
 1.5000e+03 + 4.3000e+03i

>> cumsum(a),cumsum(b)
ans =
 6 10 12 23 32 39

ans =
 9.0000 + 2.0000i 8.0000 + 5.0000i 8.0000 + 10.0000i 2.0000 + 5.0000i 6.0000
+ 5.0000i

>> cumprod(a),cumprod(b)
ans =
 6 24 48 528 4752 33264

ans =
 1.0e+03 *

```



0.0090 + 0.0020i      -0.0150 + 0.0250i      -0.1250 - 0.0750i      0.3750 + 1.0750i      1.5000  
+ 4.3000i

### 1.8.5. Definición de matrices o arreglos bidimensionales

Para definir una matriz no hace falta conocer su tamaño, se determinan el número de filas y de columnas en función del número de elementos que se introducen.

Las matrices se definen por filas, estando los elementos de una misma fila separados por espacios en blanco o comas, mientras que las filas están separadas por punto y coma (;).

#### *Ejemplos:*

Existen diferentes formas de crear una matriz:

```
>> A=[1 2 3; 4 5 6; 7 8 9]
```

```
A=
 1 2 3
 4 5 6
 7 8 9
```

En este ejemplo los elementos de las filas están separados por espacios en blanco.

```
>> B=[1,2,3,4;5,6,7,8]
```

```
B=
 1 2 3 4
 5 6 7 8
```

Para ingresar una matriz se puede utilizar la coma para separar los elementos de las filas.

```
>> C=[1 2;
 3 4;
 5 6;
 7 8]
```

```
C=
 1 2
 3 4
 5 6
 7 8
```

```
>> D=[1,2;
 3,4;
 5,6;
 7,8]
```

```
D=
 1 2
 3 4
 5 6
 7 8
```

Se puede definir la matriz utilizando una línea para los elementos de cada fila ya sea separado por espacios o por comas, la instrucción se termina solo cuando se cierra los corchetes.

```
>> R=[5 6 7 8 9
 3 4 5 6 1]
```

```
9 6 7 4 2]
```

```
R =
 5 6 7 8 9
 3 4 5 6 1
 9 6 7 4 2
```

Si se utiliza una línea para definir cada fila de la matriz se puede omitir el uso del punto y coma para crear la siguiente fila.

De la misma manera para generar las filas de la matriz se puede utilizar la creación de vectores de forma rápida que se revisó anteriormente para no tener que escribirlos uno a uno.

```
>> A=[11:15;0:-1:-4;linspace(20,30,5)]

A =
 11.00 12.00 13.00 14.00 15.00
 0.00 -1.00 -2.00 -3.00 -4.00
 20.00 22.50 25.00 27.50 30.00
```

### 1.8.6. Funciones para crear matrices

**Función rand:** crea una matriz de números aleatorios distribuidos uniformemente entre 0 y 1. **Función randn:** crea una matriz de números aleatorios distribuidos normalmente. **Función randi:** crea una matriz de enteros aleatorios distribuidos uniformemente. En el primer argumento se proporciona el valor máximo, y a continuación el tamaño de la matriz que desea crear.

#### *Ejemplos:*

- Crear un vector de 5 por 1 de números aleatorios distribuidos uniformemente con el nombre y1.

```
>> y1=rand(5,1)
y1 =
 0.4709
 0.2305
 0.8443
 0.1948
 0.2259
```

- Cree un vector de 5 por 1 de números aleatorios normalmente distribuidos con el nombre y2 usando la función randn.

```
>> y2=randn(5,1)
y2 =
 -0.7120
 -1.1742
 -0.1922
 -0.2741
 1.5301
```

- Cree un vector de 5 por 1 de enteros aleatorios distribuidos uniformemente con un valor máximo de 10. Nombre el vector y3.

```
>> y3=randi(10,5,1)
y3 =
```

7  
8  
3  
2  
3

### 1.8.7. Creación y concatenación de matrices

En MATLAB se puede crear matrices en términos de otra matriz que ya se haya definido. Para crear la matriz se puede utilizar cualquiera de las formas ya indicadas.

#### *Ejemplos:*

1. >> A=[4.5 3.2 1.5]

A =  
4.50 3.20 1.50

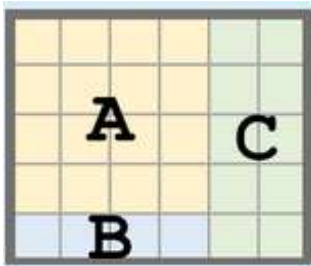
>> B=[ 3.7 A]

B =  
3.70 4.50 3.20 1.50

>> C=[9.5 3.4 2.1 5.6;B]

C =  
9.50 3.40 2.10 5.60  
3.70 4.50 3.20 1.50

2. Crear una matriz llamada F con los elementos que forman partes de la configuración que se muestra a continuación



```
>> A=rand(4,4);
>> B=randi(10,1,4);
>> C=randn(5,2);
>> F=[[A;B],C]
```

F =  
0.3342 0.7441 0.6099 0.5767 0.8617 -2.1924  
0.6987 0.5000 0.6177 0.1829 0.0012 -2.3193  
0.1978 0.4799 0.8594 0.2399 -0.0708 0.0799

|        |        |        |         |         |         |
|--------|--------|--------|---------|---------|---------|
| 0.0305 | 0.9047 | 0.8055 | 0.8865  | -2.4863 | -0.9485 |
| 1.0000 | 5.0000 | 2.0000 | 10.0000 | 0.5812  | 0.4115  |

Para acceder a los elementos de una matriz se utiliza números enteros llamados índices. Lo podemos hacer al igual que los vectores de dos formas:

Con un solo índice los elementos se enumeran de columna en columna.

```
>> T=[5 , 6 , 7; 1, 8 , 3; 5 , 4 , 2]
>> T(4)
ans
 6
```

O podemos hacerlo con el número de la fila y la columna que corresponde

```
>>T(3,1)
ans
 5
```

**Indexación en un arreglo:** Se puede cambiar los valores en una matriz, o incluir valores adicionales, con un número índice para especificar un elemento particular.

```
>> T(2)=-10
T =
 5.00 6.00 7.00
 -10.00 8.00 3.00
 5.00 4.00 2.00
```

```
>> T(3,2)=-10
T =
 5.00 6.00 7.00
 -10.00 8.00 3.00
 5.00 -10.00 2.00
```

### 1.8.8. Uso del operador dos puntos

El operador dos puntos es un operador que se lo utiliza para definir nuevos vectores o matrices de forma rápida así como modificar las existentes.

El operador dos puntos también se lo utiliza para extraer datos de las matrices, una característica que es muy útil en análisis de datos.

Si se coloca : en el primer índice indica que se desea extraer toda o todas las filas, y si se coloca en el segundo índice indica toda la columna o las columnas.

**Ejemplos:**

```
>> M=[1 2 3 4 5 6 7 8 9; 2 3 4 5 6 7 8 9 2;3 4 5 6 7 8 9 8 1; 5 7 1 8 2 4 6 8 3;6 8 9 1 3 4 5
7 6]
M =
 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00
 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 2.00
 3.00 4.00 5.00 6.00 7.00 8.00 9.00 8.00 1.00
 5.00 7.00 1.00 8.00 2.00 4.00 6.00 8.00 3.00
 6.00 8.00 9.00 1.00 3.00 4.00 5.00 7.00 6.00
```

1. Extraer todos los elementos de la columna 1

```
>> x=M(:,1)
x =
 1.00
 2.00
 3.00
 5.00
 6.00
```

2. Extraer todos elementos de la fila 3

```
>> x=M(3,:)
x =
 3.00 4.00 5.00 6.00 7.00 8.00 9.00 8.00 1.00
```

3. Extraer los elementos de las columnas 3 hasta la 5

```
>> x=M(:,3:5)
x =
 3.00 4.00 5.00
 4.00 5.00 6.00
 5.00 6.00 7.00
 1.00 8.00 2.00
 9.00 1.00 3.00
```

4. Extraer todos los elementos de las filas 2 hasta la 4.

```
>> x=M(2:4,:)
x =
 2 3 4 5 6 7 8 9 2
 3 4 5 6 7 8 9 8 1
 5 7 1 8 2 4 6 8 3
```

Si se utiliza el operador : con un valor inicial y un valor final se da por entendido que el intervalo es de 1 en 1.

5. Extraer todos los elementos de las columnas pares.

```
>> x=M(:,2:2:9)
x =
 2 4 6 8
 3 5 7 9
 4 6 8 8
 7 8 4 8
 8 1 4 7
```

6. Extraer todos los elementos de las filas impares.

```
>> x=M(1:2:5,:)
```

```
x =
 1 2 3 4 5 6 7 8 9
 3 4 5 6 7 8 9 8 1
 6 8 9 1 3 4 5 7 6
```

En el ejercicio 5 y 6 se indica el valor inicial:intervalo:valor final, el intervalo que se utiliza es de 2 en 2

7. Extraer todos los elementos de las columnas 1-2-5-7-8

```
>> x=M(:,[1,2,5,7,8])
x =
 1 2 5 7 8
 2 3 6 8 9
 3 4 7 9 8
 5 7 2 6 8
 6 8 3 5 7
```

Para extraer filas o columnas que no tienen un intervalo regular utilizamos [], se en lista solamente el numero de las columnas a extraer.

8. Extraer todos los elementos de las fila 1-2-5.

```
>> x=M([1,2,5],:)
x =
 1 2 3 4 5 6 7 8 9
 2 3 4 5 6 7 8 9 2
 6 8 9 1 3 4 5 7 6
```

9. Extraer los elementos que se encuentran de la fila 2 a la 4 y de las columnas 4 hasta la 7.

```
>> x=M(2:4,4:7)
x =
 5 6 7 8
 6 7 8 9
 8 2 4 6
```

10. Extraer el último elemento de la fila 3.

```
>> x=M(3,end)
x =
 1
```

Se puede utilizar la palabra end para identificar el ultimo elemento de una fila o columna, dependiendo si se encuentra como primer índice o segundo índice.

11. Extraer el último elemento de la columna 5.

```
>> x=M(end,5)
x =
 3
```

Para extraer el ultimo elemento de la matriz se puede utilizar como indices (end , end) o a su vez solo (end).

12. Extraer el último elemento de la matriz.

```
>> x=M(end,end)
x =
 6
>> x=M(end)
x =
 6
```

Se puede convertir la matriz en un vector tipo columna usando el operador `:`.

13. Extraer de la matriz M las filas de la 3:5 las columnas de la 6:8 y almacene en x, luego convertir la matriz x resultante en un vector tipo de columna y almacene en s.

```
>> x=M(3:5,6:8)
x =
 8 9 8
 4 6 8
 4 5 7

>> s=x(:)
s =
 8
 4
 4
 9
 6
 5
 8
 8
 7
```

### 1.8.9. Características de la matriz

Para conocer las características de una matriz, podemos utilizar los siguientes comandos:

**[f c]=size(A):** Calcula la cantidad de filas y columnas que tiene la matriz A, y el resultado lo almacena en f y c respectivamente.

**det(A):** Calcula el determinante de la matriz A.

**rank(A):** Halla el rango de una matriz A.

**trace(A):** Muestra la traza de la matriz A.

### 1.8.10. Matrices especiales

▪ **eye(n):** Devuelve la matriz identidad de orden n.

▪ **zeros(n,m):** Devuelve la matriz nula de dimensiones nxm.

- **rand(n)**- **rand(n,m)**: Devuelve una matriz aleatoria uniforme de orden n (o nxm), con valores entre 0 y 1.
- **randn(n)**- **randn(n,m)**: Devuelve una matriz aleatoria normal de orden n (o nxm).
- **ones(n,m)**: Devuelve una matriz de dimensiones nxm con todos sus elementos iguales a 1.
- **A'**: Muestra la transpuesta de la matriz A.
- **inv(A)**: Devuelve la matriz inversa de una matriz A no singular (invertible).
- **diag(v)**: Devuelve una matriz diagonal con los elementos del vector v.
- **diag(A)**: Extrae la diagonal de la matriz A como un vector columna.
- **tril(A)**: Devuelve la parte triangular inferior de una matriz A.
- **triu(A)**: Devuelve la parte triangular superior de una matriz A.

**Ejemplos:**

```
>> eye(4)
ans =
 1 0 0 0
 0 1 0 0
 0 0 1 0
 0 0 0 1

>> zeros(3,2)
ans =
 0 0
 0 0
 0 0

>> ones(2,3)
ans =
 1 1 1
 1 1 1

>> rand(3)
ans =
 0.6324 0.5469 0.1576
 0.0975 0.9575 0.9706
 0.2785 0.9649 0.9572

>> randn(3,2)
ans =
 1.4172 0.7172
 0.6715 1.6302
 -1.2075 0.4889
```



### 1.8.11. Funciones

#### Trigonométricas:

Funciones elementales que admiten como argumento una matriz compleja  $Z$ .

- $\sin(\mathbf{z})$ : Función seno
- $\sinh(\mathbf{Z})$ : Función seno hiperbólico
- $\text{asin}(\mathbf{Z})$ : Función arcoseno
- $\text{asinh}(\mathbf{Z})$ : Función arcoseno hiperbólico
- $\cos(\mathbf{Z})$ : Función coseno
- $\cosh(\mathbf{Z})$ : Función coseno hiperbólico
- $\text{acos}(\mathbf{Z})$ : Función arcocoseno
- $\text{acosh}(\mathbf{Z})$ : Función arcocoseno hiperbólico
- $\tan(\mathbf{Z})$ : Función tangente
- $\tanh(\mathbf{Z})$ : Función tangente hiperbólica
- $\text{atan}(\mathbf{Z})$ : Función arcotangente
- $\text{atanh}(\mathbf{Z})$ : Función arcotangente hiperbólica
- $\cot(\mathbf{Z})$ : Función cotangente
- $\coth(\mathbf{Z})$ : Función cotangente hiperbólica
- $\text{acot}(\mathbf{Z})$ : Función arcocotangente
- $\text{acoth}(\mathbf{Z})$ : Función arcocotangente hiperbólica
- $\sec(\mathbf{Z})$ : Función secante
- $\text{sech}(\mathbf{Z})$ : Función secante hiperbólica
- $\text{asec}(\mathbf{Z})$ : Función arcosecante
- $\text{asech}(\mathbf{Z})$ : Función arcosecante hiperbólica

- **csc(Z)**: Función cosecante
- **csch(Z)**: Función cosecante hiperbólica
- **acsc(Z)**: Función arcocosecante
- **acsch(Z)**: Función arcocosecante hiperbólica

### Exponenciales

- **exp(z)**: Función exponencial de base e
- **log(Z)**: Función logaritmo neperiano
- **log10(Z)**: Función logaritmo decimal
- **sqrt(Z)**: Función raíz cuadrada

### Complejas

- **abs(z)**: Modulo o valor absoluto
- **angle(Z)**: Argumento
- **conj(Z)**: Complejo conjugado
- **imag(Z)**: Parte imaginaria
- **real(Z)**: Parte real

### Numéricas

- **fix(z)**: Elimina las partes decimales
- **floor(Z)**: Redondea los decimales al menor entero más cercano
- **ceil(Z)**: Redondea los decimales al mayor entero más cercano
- **round(Z)**: Efectúa el redondeo común de decimales

## 1.9. Ejercicios propuestos

1. Crear un vector de fila llamado  $x$  que comienza en 0, termina en 1 y contiene 80 elementos espaciados uniformemente.
2. Defina una variable llamada  $d$  que contenga un valor entre 1 y 5. Use esta variable como índice para extraer el elemento del vector  $x=[19\ 0.8\ 1.51\ 3.87\ 10]$ , almacene el resultado en la variable  $p$ .
3. Crear un vector de 5 a 20 con incremento de 4.  
Eleva al cubo cada elemento.  
Sumar de forma acumulada sus elementos.  
Mostrar el resultado del producto de sus elementos
4. Crear un vector  $p$  compuesto por 60 elementos espaciados en 5 unidades con elemento inicial 0. En el vector  $r$  guardar los elementos del vector  $p$  que van desde la posición 20 a la 10 decreciendo en 3 posiciones por elemento y desde la 30 a 60 con incremento de 2.  
Defina un vector complejo  $s$  con 5 elementos, y halle la suma y productos acumulados del vector. Para el vector  $s$ , encuentre el mayor y menor de sus elementos.
5. Genere un vector de 12 elementos que corresponde a los gastos mensuales de consumo de electricidad y determine el gasto promedio de consumo en el año.
6. Encontrar los vectores  $c1$  y  $c2$  que representan las evaluaciones de las funciones coseno y tangente.
7. Subdividir los elementos de  $c1$  de forma que las posiciones impares conformen el vector  $s1$  y las posiciones pares el vector  $s2$ .
8. Mostrar en  $s3$  el cuadrado de los elementos de la suma de  $s1$  con  $s2$ .
9. Crea las matrices  $A$  y  $B$  de orden 3

$$A = \begin{bmatrix} 5 & 3 & 5 \\ -7 & 9 & 1 \\ 1 & -2 & 6 \end{bmatrix}$$

$$B = \begin{bmatrix} 18 & 16 & 14 \\ 12 & 10 & 8 \\ 6 & 4 & 2 \end{bmatrix}$$

10. Multiplica la transpuesta del producto de  $A$  con  $B$ , con la inversa de  $A$ .
11. A la matriz resultante  $C$ , calcule su traza y determinante.
12. Divide a la matriz  $C$  por derecha la matriz  $A$ .

# 2

## *Introducción a la programación en Matlab*





## 2.1. Tipos de archivos de Matlab

### 2.1.1. Script

MATLAB trabaja con un poderoso lenguaje de programación, en el que se puede crear y guardar código al que se le denomina script y se lo define mediante un archivo con extensión `.m`, formado por un conjunto de instrucciones o sentencias.

Para crear un script vamos a la pestaña: Home → New → Script



Figura 2.1: Nuevo script

Para guardar un archivo `-m` se lo debe almacenar en el directorio actual, para el nombre del script sera necesario tomar en cuentas las siguientes consideraciones:

- Debe iniciar con una letra.
- Solo puede contener números, letras y el guion bajo. Ningún otro carácter especial.
- No se permite espacios en blanco
- No se debe usar palabras reservadas en el nombre del archivo.

### 2.1.2. Function

Una funcion o función se define mediante archivo con extensión `.m`, cuyo nombre debe ser el mismo nombre de la función. La primera línea del archivo inicia con la palabra `function`, y el icono del mismo se diferencia de un script al tener un `fx`.

Una función es un conjunto de instrucciones cuyo fin es realizar una tarea particular. Puede recibir uno o varios valores de entrada ( `input1` , `input2` , ... ) y producir como salida uno o varios valores ( `output1` , `output2` , ... ).

## 2.2. Entradas y salidas controladas por el usuario

Matlab tiene funciones internas que permiten al usuario ingresar valores.

```

1 function [outputArg1,outputArg2] = untitleo(inputArg1,inputArg2)
2
3 % FUNCTION Summary of this function goes here
4
5 % Detailed explanation goes here
6
7 outputArg1 = inputArg1;
8 outputArg2 = inputArg2;
9
10 end

```

Figura 2.2: Archivo de una función

### 2.2.1. Función input

Muestra en la ventana de comandos una cadena de texto que le permite al usuario proporcionar la entrada solicitada.

**Sintaxis:**

```
variable=input('Cadena de texto ');
```

**Ejemplo:**

```

>> z=input('Ingrese una valor: ')
Ingrese una valor: 5
z =
 5

```

Para ingresar una matriz usamos los corchetes [ ].

**Ejemplos:**

```

>> v=input('Ingrese un vector: ')
Ingrese un vector: [3 4 5 7]
v =
 3 4 5 7

```

```

>> m=input('Ingrese una matriz: ')
Ingrese una matriz: [2 3;4 5]
m =
 2 3
 4 5

```

Para ingresar una cadena se debe usar ' ' o especificar que es una cadena con el carácter 's', luego del mensaje.

**Ejemplos:**

```

>> n=input('Ingrese su nombre: ')
Ingrese su nombre: 'Fernando'
n =
 'Fernando'

```

```

>> n=input('Ingrese su nombre: ','s')
Ingrese su nombre: Fernando
n =
 'Fernando'

```

### 2.2.2. Opciones de Salida

Matlab tiene funciones para desplegar los resultados, a continuación se presenta dos enfoques:

- **Función de despliegue (disp):** Muestra el valor de la variable sin imprimir el nombre de la variable.

**Ejemplos:**

```
>> A=[45 98];
>> disp(A)
 45 98

>> s='Programar con Matlab'
>> disp(s)
Programar con Matlab

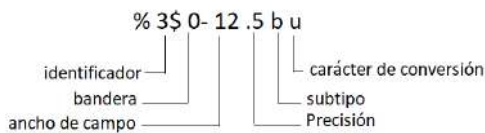
>> nombre = 'Patricia';
>> edad = 12;
>> A = [nombre,' tendrá ',num2str(edad),' este año.'];
>> disp(A)
Patricia tendrá 12 este año.
```

En el ejemplo se muestra varias variables en la misma instrucción concatenadas con el operador []. Con la función num2str se convierte los valores numéricos en caracteres y se muestra los resultados en la ventana de comandos con disp.

- **Función impresa formateada fprintf:** se utiliza para especificar el formato a usar al desplegar los valores.

```
fprintf(text) Muestra texto formateado.
fprintf(formato,var) Formatea var como se especifica en formato.
```

Un operador de formato comienza con un signo de porcentaje, %, y termina con un carácter de conversión el cual es obligatorio. Se puede especificar operadores de identificador, banderas, ancho de campo, precisión y subtipo entre % y el carácter de conversión sin espacios.



**Figura 2.3:** Formateo de la salida fprintf

**Tabla 2.1:** Caracteres de conversión para formatear datos numéricos y de caracteres

| Tipo                     | Conversión | Detalles                                                      |
|--------------------------|------------|---------------------------------------------------------------|
| Entero                   | %d         | Sin decimales                                                 |
| Número de punto flotante | %f         | Se puede usar un operador de precisión                        |
|                          | %e         | y especificar el número de dígitos después del punto decimal. |
| Carácter o cadena        | %g         | Es más compacto de %e o %f, sin ceros finales.                |
|                          | %c         | Carácter simple                                               |
|                          | %s         | Vector de caracteres o matriz de cadenas.                     |

En la instrucción se puede incluir comandos de formato especial las cuales se especifican en la siguiente tabla.



**Tabla 2.2:** Comandos de formato especial

| Comando | Acción                |
|---------|-----------------------|
| \n      | Salto de línea        |
| \r      | regreso de carro      |
| \t      | tabulador             |
| \b      | retroceder un espacio |

**Ejemplos:**

```
>> a=5;
>> fprintf('El valor de la variable a es %d \n',a)
El valor de la variable a es 5
```

```
>> a=5;b=4.9;
>> c=a+b;
>> fprintf('La suma de %d mas %f es %f \n',a,b,c)
La suma de 5 mas 4.900000 es 9.900000
```

En este ejemplo se muestra la variable a que es un numero entero se imprime usando la letra d y las variables b y c se usa la letra f para números decimales se imprime con 6 dígitos decimales.

```
>> fprintf('La suma de %d mas %2.2f es %2.3f \n',a,b,c)
La suma de 5 mas 4.90 es 9.900
```

Se puede definir el ancho del campo y la cantidad de dígitos decimales con la letra f, para la variable b se especifica 2 como ancho de campo y 2 dígitos decimales, para la variable c ancho de campo 2 y 3 dígitos decimales.

## 2.3. Tipos de Estructuras

- **Secuenciales:** o programación secuencial, son instrucciones o listas de comandos que se ejecutan una después de otra.
- **Selección:** o bifurcación o condición son estructuras que permiten seleccionar a través de una condición una de dos alternativas posibles cuando está es verdadera o falsa, ejecutándose procesos diferentes. Para la condición se utiliza operadores de relación y lógicos.

Existen diferentes tipos de estructuras de condición o bifurcación:

- Bifurcación Simple.
  - Bifurcación Anidada
  - Bifurcación Compuesta
  - Bifurcacion Múltiple
- **Repetición:** o bucles permiten repetir un número determinado de veces un conjunto de sentencias. El numero de repeticiones depende de un contador o de la evaluación de una condición lógica.

Entre las estructuras de repetición podemos mencionar:

- Estructura While
- Estructura for

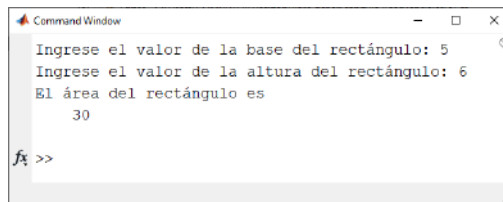
## 2.4. Estructuras secuenciales

Las estructuras secuenciales son aquellas en las que una instrucción sigue a otra en secuencia. Las instrucciones se ejecutan de tal modo que la salida de una es la entrada de la siguiente y así hasta el fin del proceso.

### Ejemplos:

1. Realizar un script que determine el área de un rectángulo.

```
1 clear
2 clc
3 b=input('Ingrese el valor de la base del rectángulo: ');
4 h=input('Ingrese el valor de la altura del rectángulo: ');
5 a=b*h;
6 disp('El area del rectangulo es')
7 disp(a)
```

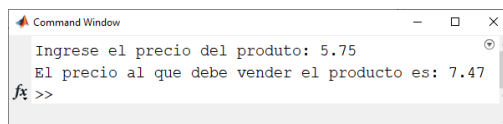


```
Command Window
Ingrese el valor de la base del rectángulo: 5
Ingrese el valor de la altura del rectángulo: 6
El área del rectángulo es
30
fx >>
```

Figura 2.4: Salida ejemplo 1 estructuras secuenciales

2. Un cliente compra un producto a un precio determinado y desea venderlo para obtener una ganancia del 30%. Realizar un script para calcular el precio final al que lo debe vender.

```
1 clear
2 clc
3 p=input('Ingrese el precio del producto: ');
4 g=p*30/100;
5 pv=p+g;
6 fprintf('El precio al que debe vender el producto es: %.2f \n',pv)
```

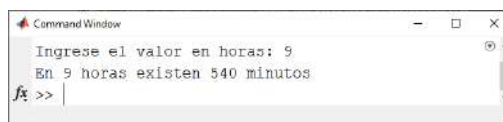


```
Command Window
Ingrese el precio del producto: 5.75
El precio al que debe vender el producto es: 7.47
fx >>
```

Figura 2.5: Salida ejemplo 2 estructuras secuenciales

3. Realizar un script que transforme una cantidad h dada en horas a minutos.

```
1 h=input('Ingrese el valor en horas: ');
2 m=h*60;
3 fprintf('En %d horas existen %d minutos \n',h,m)
```

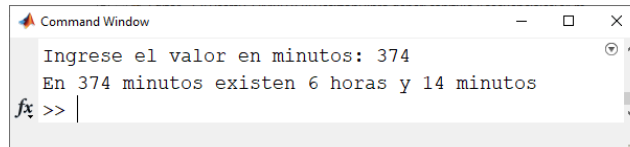


```
Command Window
Ingrese el valor en horas: 9
En 9 horas existen 540 minutos
fx >>
```

Figura 2.6: Salida ejemplo 3 estructuras secuenciales

4. Realizar un script que lea una cantidad  $m$  dada en minutos deseando transformar a horas y minutos.

```
1 m=input('Ingrese el valor en minutos: ');
2 h=floor(m/60);
3 m1=mod(m,60);
4 fprintf('En %d minutos existen %d horas y %d minutos \n',m,h,m1)
```

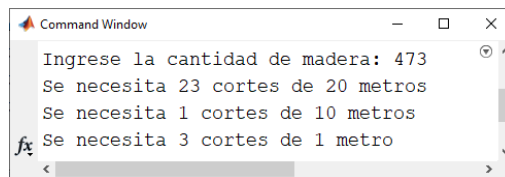


```
Command Window
Ingrese el valor en minutos: 374
En 374 minutos existen 6 horas y 14 minutos
fx >> |
```

Figura 2.7: Salida ejemplo 4 estructuras secuenciales

5. En una empresa se fábrica madera Mdf en cortes de 20, 10 Y 1 metro, se requiere determinar los cortes necesarios de esta madera para entregar  $X$  metros a un cliente.

```
1 X=input('Ingrese la cantidad de madera: ');
2 C20=floor(X/20);
3 Z=mod(X,20);
4 C10 =floor(Z/10);
5 C1=mod(Z,10);
6 fprintf('Se necesita %d cortes de 20 metros \n',C20)
7 fprintf('Se necesita %d cortes de 10 metros \n',C10)
8 fprintf('Se necesita %d cortes de 1 metro \n',C1)
```

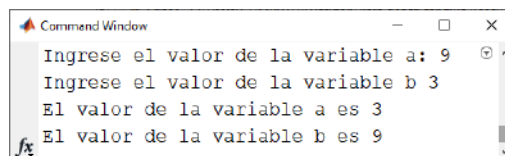


```
Command Window
Ingrese la cantidad de madera: 473
Se necesita 23 cortes de 20 metros
Se necesita 1 cortes de 10 metros
fx Se necesita 3 cortes de 1 metro
```

Figura 2.8: Salida ejemplo 5 estructuras secuenciales

6. Realizar un script que lea 2 valores, y como salida imprima los 2 valores cambiado de variable.

```
1 a=input('Ingrese el valor de la variable a: ');
2 b=input('Ingrese el valor de la variable b ');
3 c=a;
4 a=b;
5 b=c;
6 fprintf('El valor de la variable a es %d \n',a)
7 fprintf('El valor de la variable b es %d \n',b)
```



```
Command Window
Ingrese el valor de la variable a: 9
Ingrese el valor de la variable b 3
El valor de la variable a es 3
fx El valor de la variable b es 9
```

Figura 2.9: Salida ejemplo 6 estructuras secuenciales

7. Realizar un script que descomponga un número de 3 dígitos, suponer que el usuario solo va a ingresar números de 3 dígitos.

```

1 clear
2 clc
3 N=input('Ingrese un numero de 3 cifras: ');
4 C=floor(N/100);
5 M=mod(N,100);
6 D=floor(M/10);
7 U=mod(M,10);
8 fprintf('Primer digito %d \n',C)
9 fprintf('Segundo digito %d \n',D)
10 fprintf('Tercer digito %d \n',U)

```

```

Command Window
Ingrese un número de 3 cifras: 593
Primer digito 5
Segundo digito 9
Tercer digito 3
fx >>

```

Figura 2.10: Salida ejemplo 7 estructuras secuenciales

8. Una empresa que produce prendas de vestir le paga a sus profesores 13.5 dólares la hora y le hace un descuento del 3% por concepto de caja de ahorro. Calcule el monto del descuento y el monto mensual a pagar al trabajador, sabiendo que trabaja los 5 días a la semana las 8 horas diarias.

```

1 clear
2 clc
3 pd=13.5*8; %precio por hora por 8 horas diarias
4 pm=pd*30; %valor diario por 30 dias
5 fprintf('El valor mensual a pagar al trabajador es %.2f \n', pm)

```

```

Command Window
El valor mensual a pagar al trabajador es 3240.00
fx >>

```

Figura 2.11: Salida ejemplo 8 estructuras secuenciales

9. Realizar un script que permita calcular el descuento y el monto a pagar por un medicamento en una farmacia que ofrece un descuento del 35%.

```

1 clear
2 clc
3 precio=input('Ingrese el precio del medicamento: ');
4 des=precio*35/100;
5 vp=precio - des;
6 fprintf('El valor del descuento es %.2f \n',des)
7 fprintf('El valor a pagar por el medicamento es %.2f \n',vp)

```

```

Command Window
Ingrese el precio del medicamento: 14.75
El valor del descuento es 5.16
El valor a pagar por el medicamento es 9.59
fx >>

```

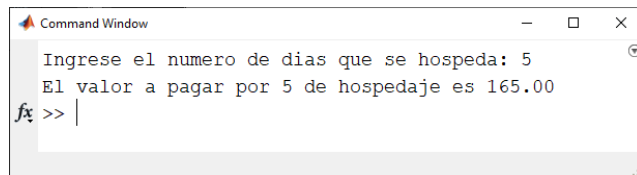
Figura 2.12: Salida ejemplo 9 estructuras secuenciales

10. Un Hotel tiene una promoción para sus clientes, cobra por una habitación 45 dólares el primer día, y por el resto de los días cobra 30 dólares diarios. Realizar un script que determine el monto a pagar por la habitación si se hospedo por varios días.

```

1 clear
2 clc
3 dias=input('Ingrese el numero de dias que se hospeda: ');
4 s=dias-1;
5 v=(s*30)+45;
6 fprintf('El valor a pagar por %d de hospedaje es %.2f \n',dias,v)

```



**Figura 2.13:** Salida ejemplo 10 estructuras secuenciales

### 2.4.1. Ejercicios propuestos estructuras secuenciales

1. Una inmobiliaria de la ciudad vende terrenos a 120 dólares el metro cuadrado. El comprador debe dar un anticipo y el resto pagar en 24 cuotas. Realizar un script que determine el monto de cada cuota.
2. Realizar un script que calcule la distancia entre dos puntos en el plano cartesiano, para lo cual debe ingresar los datos del par ordenado.
3. Realizar un script que descomponga un número de 3 dígitos y muestre un nuevo número con los dígitos del número de atrás hacia adelante. Suponer que el usuario solo ingresa números de 3 dígitos.
4. Una empresa le hace los siguientes descuentos sobre el sueldo a sus trabajadores: 2% por Seguro privado, 0,5% por extension del seguro a sus hijos y 1% para la caja chica de la empresa. Realizar un script que determine el monto de los descuentos y el monto que se le va a pagar al trabajador.
5. Realizar un script en el que se ingrese el tiempo de procese de un producto en horas, minutos y segundos. Calcule el costo total del proceso del producto si se conoce que el precio por segundo es de 0.30 centavos.

## 2.5. Estructuras de Control

Una estructura de control es una instrucción MATLAB que permite tomar decisiones sobre una condición, de tal manera que se ejecuta un grupo de instrucciones si la condición se cumple o no.

**Tabla 2.3:** Tipos de condiciones

| Nombre                | Definición                                                                                                               |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------|
| bifurcación simple    | Si la condición (una expresión lógica) es verdadera, se ejecutan las sentencias y sigue a end.                           |
| bifurcación anidada   | incluye dos sentencias condicionales, lo que hace posible ejecutar uno de entre tres grupos de instrucciones diferentes. |
| bifurcación compuesto | En un mismo bloque se puede tener mas de una condición                                                                   |
| bifurcación múltiple  | existe una serie de opciones de ruta de programación para una variable dada, dependiendo de su valor.                    |

### 2.5.1. Bifurcación simple

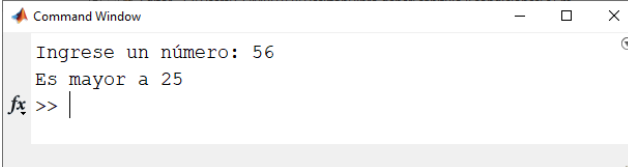
En una bifurcación simple se ejecutan instrucciones si se cumple la condición. Su sintaxis es:

```
if condición
 instrucciones (Se ejecuta si la condición es verdadera)
end
```

#### *Ejemplos:*

1. Realizar un script que determine si un numero es mayor que 25.

```
1 clear
2 clc
3 N=input('Ingrese un numero: ');
4 if N>25
5 disp('Es mayor a 25')
6 end
```

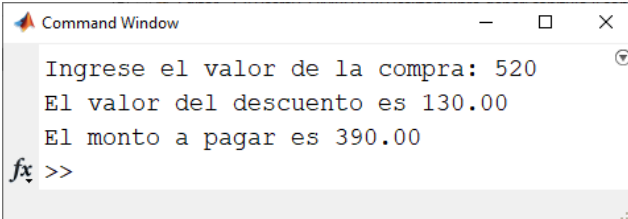


```
Command Window
Ingrese un número: 56
Es mayor a 25
fx >> |
```

**Figura 2.14:** Salida ejemplo 1 estructuras de bifurcación

2. Realizar un script que permita determinar el 25 % de descuento que recibirá un cliente cuando el monto a pagar supere los 450 dólares. Muestre el descuento y el monto total a pagar por el cliente.

```
1 clear
2 clc
3 c=input('Ingrese el valor de la compra: ');
4 if c>450
5 d=c*25/100;
6 c=c-d;
7 fprintf('El valor del descuento es %.2f \n',d)
8 end
9 fprintf('El monto a pagar es %.2f \n',c)
```



```
Command Window
Ingrese el valor de la compra: 520
El valor del descuento es 130.00
El monto a pagar es 390.00
fx >>
```

**Figura 2.15:** Salida ejemplo 2 estructuras de bifurcación

En una forma más general se ejecutan unos comandos si la condición es cierta, y otros si la condición es falsa. Su sintaxis es:

```
if condición
 instrucciones1 (se ejecutan si la condición es verdadera)
else
 instrucciones2 (se ejecutan cuando la condición no se cumple)
end
```

- Realizar un script que determine si un número ingresado es positivo o negativo, considerando al cero como positivo.

```

1 clear
2 clc
3 N=input('Ingrese un numero: ');
4 if N>=0
5 disp('ES POSITIVO')
6 else
7 disp('ES NEGATIVO')
8 end

```

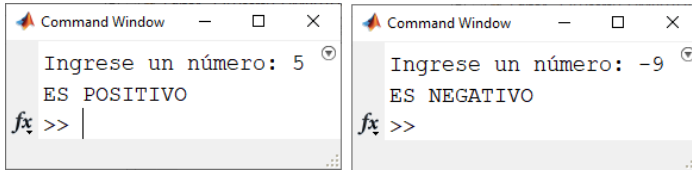


Figura 2.16: Salida ejemplo 3 estructuras de bifurcación

- Realizar un script que determine si un número ingresado es par o impar.

```

1 clear
2 clc
3 X=input('Ingrese un numero: ');
4 R=mod(X,2);
5 if R==0
6 disp('ES PAR')
7 else
8 disp('ES IMPAR')
9 end

```

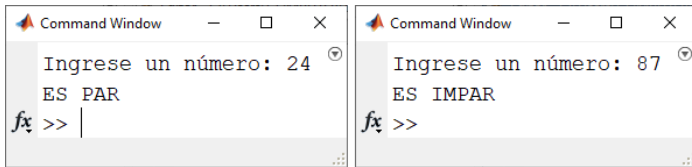


Figura 2.17: Salida ejemplo 4 estructuras de bifurcación

- Realizar un script que determine el costo de un pedido, si se conoce la cantidad de artículos pedidos y el precio unitario. Si la cantidad pedida excede las 30 unidades se hace un descuento del 15%.

```

1 clear
2 clc
3 u=input('Ingrese la cantidad de articulos pedidos: ');
4 p=input('Ingrese el precio unitario del articulo: ');
5 st=u*p;
6 if u>30
7 d=st*15/100;
8 else
9 d=0;
10 end
11 mp=st-d;
12 fprintf('El valor del descuento es %.2f \n',d)
13 fprintf('El monto a pagar es %.2f \n',mp)

```

```

Command Window
Ingrese la cantidad de artículos pedidos: 35
Ingrese el precio unitario del artículo: 5
El valor del descuento es 26.25
El monto a pagar es 148.75
fx >>

```

Figura 2.18: Salida ejemplo 5 estructuras de bifurcación

6. En una boutique se tiene una promoción: a todos los vestidos que tienen un precio superior a \$150 se les aplicará un descuento de 12%, a todos los demás se les aplicará sólo 8%. Realizar un script que permita determinar el precio final que se debe pagar por comprar un vestido y de cuánto es el descuento que obtendrá.

```

1 clear
2 clc
3 c=input('Ingrese el precio del vestido: ');
4 if c>150
5 des=c*12/100;
6 else
7 des=c*8/100;
8 end
9 pf=c-des;
10 fprintf('El valor del descuento es %.2f \n',des)
11 fprintf('El monto a pagar es %.2f \n',pf)

```

The figure shows two separate Command Window windows. The top window shows the input '175', resulting in a discount of 21.00 and a total payment of 154.00. The bottom window shows the input '85', resulting in a discount of 6.80 and a total payment of 78.20.

```

Command Window
Ingrese el precio del vestido: 175
El valor del descuento es 21.00
El monto a pagar es 154.00
fx >> |

Command Window
Ingrese el precio del vestido: 85
El valor del descuento es 6.80
El monto a pagar es 78.20
fx >>

```

Figura 2.19: Salida ejemplo 6 estructuras de bifurcación

7. Un empleado desea calcular su salario semanal el cual se obtiene de la siguiente manera: Si trabaja 40 horas o menos se le paga 3.5 dólares por hora Si trabaja mas 40 horas se le paga 3.5 dólares por cada una de las 40 primeras horas y 4.5 dólares por cada hora extra.

```

1 clear
2 clc
3 h=input('Ingrese el numero de horas que trabaja el empleado: ');
4 if h≤40
5 vp=h*3.50;
6 else

```



```

7 ext=h-40;
8 ve=ext*4.5;
9 vp=(40*3.50)+ve;
10 end
11 fprintf('Por %d horas trabajadas se le debe cancelar %.2f \n',h,vp)

```

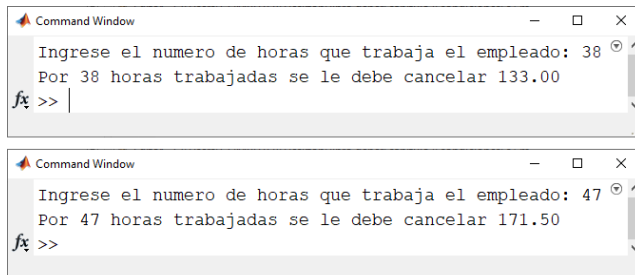


Figura 2.20: Salida ejemplo 7 estructuras de bifurcación

8. Realizar un programa que indique la relación que existe entre dos valores. No considerar la posibilidad de que sean iguales.

```

1 clear
2 clc
3 X=input('INGRESE UN NUMERO: ');
4 Y=input('INGRESE UN NUMERO: ');
5 if X>Y
6 fprintf('%d es mayor a %d \n',X,Y)
7 else
8 fprintf('%d es mayor a %d \n',Y,X)
9 end

```

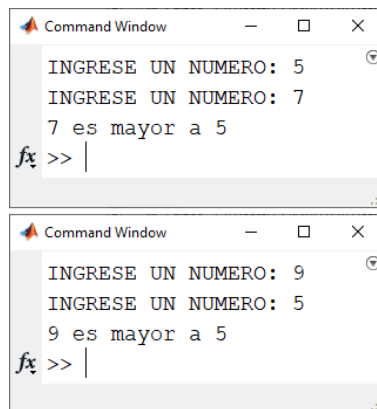


Figura 2.21: Salida ejemplo 8 estructuras de bifurcación

## 2.5.2. Bifurcación anidada

Se fundamenta en la Bifurcación simple con la diferencia que dentro de una condición existe una nueva condición, es decir dentro de una bifurcación existe como proceso o parte de este otra bifurcación simple repitiendo este proceso las veces necesarias siempre cumpliendo la regla que consiste en una entrada y una salida en cada bifurcación.

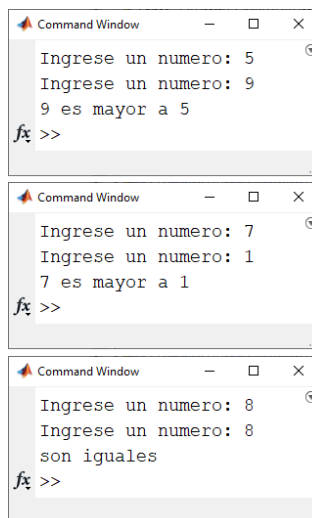
Su sintaxis es:

```
if condicion1
 comandos1 (Se ejecutan cuando la condición1 es cierta)
else
 if condición2
 comando2 (Se ejecutan cuando la condición1 no se cumple y sí la condición2).
 else
 if condición3
 comando3 (Se ejecutan cuando la condición1,condición2 no se cumplen
 y se cumple la condición3)
 else
 comandos4 (Se ejecutan cuando la condición1,condición2 y
 la condición3 no se cumplen)
 end
 end
end
...
end
```

### ***Ejemplos:***

9. Realizar un programa que indique la relación que existe entre dos valores.

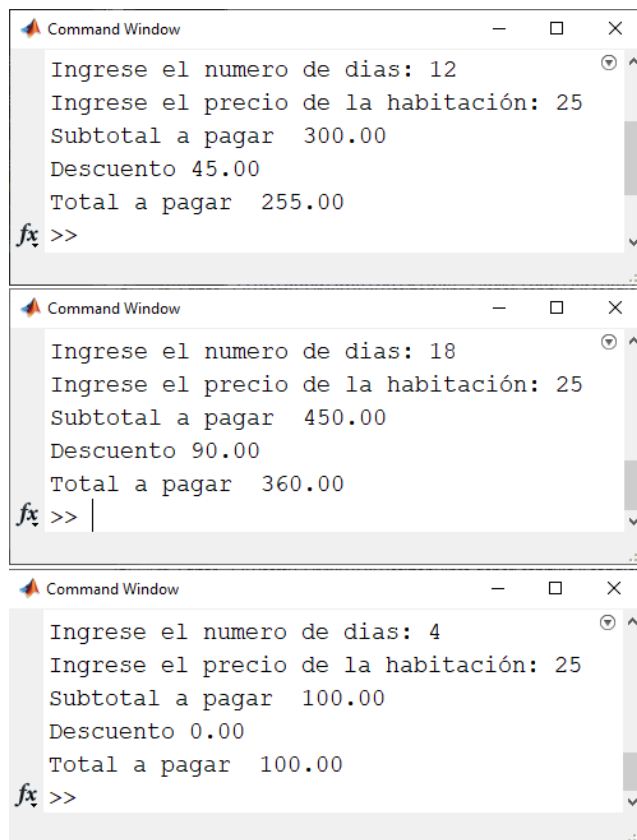
```
1 clear
2 clc
3 X=input('Ingrese un numero: ');
4 Y=input('Ingrese un numero: ');
5 if X==Y
6 disp('son iguales')
7 else
8 if X>Y
9 fprintf('%d es mayor a %d \n',X,Y)
10 else
11 fprintf('%d es mayor a %d \n',Y,X)
12 end
13 end
```



**Figura 2.22:** Salida ejemplo 9 estructuras de bifurcación

10. En un hotel se hace un descuento de 10% si el hoesped se hospeda más de 5 días, de 15% si se hospeda más de 10 días y de 20% si se hospeda más de 15 días. Realice un script que lea el número de días y el precio diario de la habitación e imprima el subtotal a pagar, el descuento y el total a pagar.

```
1 ND=input('Ingrese el numero de dias: ');
2 PH=input('Ingrese el precio de la habitacion: ');
3 SB=ND*PH;
4 if ND>15
5 DES=SB*20/100;
6 else
7 if ND>10
8 DES=SB*15/100;
9 else
10 if ND>5
11 DES=SB*10/100;
12 else
13 DES=0;
14 end
15 end
16 end
17 VP=SB-DES;
18 fprintf('Subtotal a pagar %.2f \n',SB)
19 fprintf('Descuento %.2f \n',DES)
20 fprintf('Total a pagar %.2f \n',VP)
```



The figure displays three sequential screenshots of a MATLAB Command Window. Each window shows the execution of a script that calculates the total amount to be paid for a hotel stay based on the number of days and the daily room rate. The results are as follows:

| Number of Days | Room Rate (PH) | Subtotal (SB) | Discount (DES) | Total (VP) |
|----------------|----------------|---------------|----------------|------------|
| 12             | 25             | 300.00        | 45.00          | 255.00     |
| 18             | 25             | 450.00        | 90.00          | 360.00     |
| 4              | 25             | 100.00        | 0.00           | 100.00     |

Figura 2.23: Salida ejemplo 10 estructuras de bifurcación

11. Realizar un programa en el que se ingrese la nota de un examen (un numero entero entre 0 y 10

e imprima la calificación en el siguiente formato.

”Suspendo”, si la nota es menor que 5,  
”Aprobado”, si es igual a 5 y menor a 9  
”Exonerado”, si es igual a 9 o 10

```
1 clear
2 clc
3 N=input('Ingrese la calificacion: ');
4 if N>=0
5 if N<=10
6 if N<5
7 disp('Suspendo')
8 else
9 if N<9
10 disp('Aprobado')
11 else
12 disp('Exonerado')
13 end
14 end
15 else
16 disp('Calificacion no valida')
17 end
18 else
19 disp('Calificacion no valida')
20 end
```

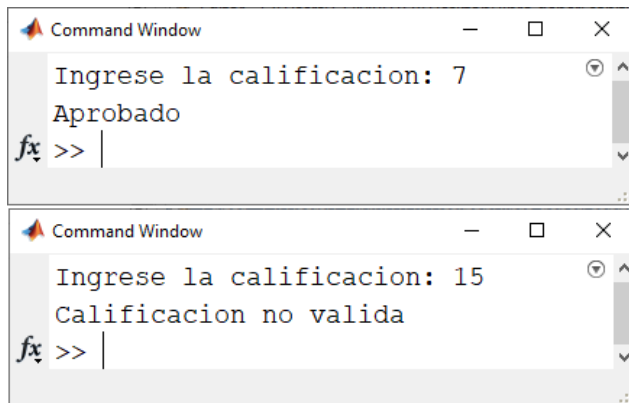


Figura 2.24: Salida ejemplo 11 estructuras de bifurcación

### Sentencia ElseIf

La sintaxis anidada anterior es equivalente, a utilizar la sentencia elseif que es mucho mas rápida al ejecutar. Cuando se anidan varios niveles de enunciados if/else suele ser difícil determinar cuales expresiones lógicas con verdaderas y cuales son falsas. ElseIf le permite comprobar múltiples criterios mientras se mantiene el código fácil de leer.

Su sintaxis es:

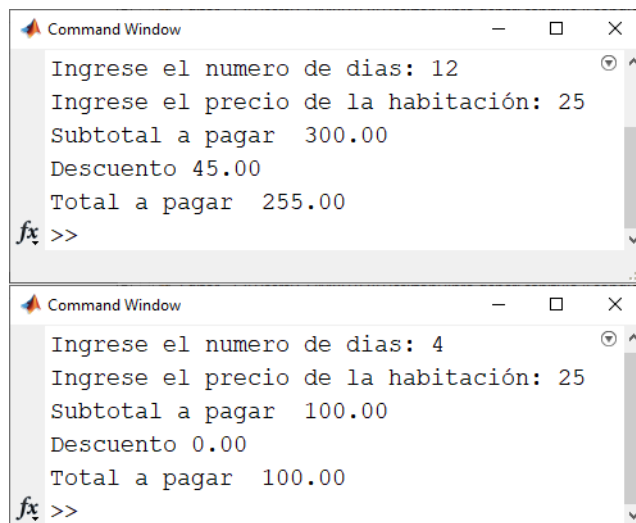
```
if condición1
 comandos1
elseif condición2
 comandos2
 elseif condición3
 comandos3
...
else
```

end

### Ejemplos:

12. Realizar el ejercicio 10 utilizando la sentencia elseif.

```
1 clear
2 clc
3 ND=input('Ingrese el numero de dias: ');
4 P=input('Ingrese el precio de la habitacion: ');
5 SB=ND*P;
6 if ND>15
7 DES=SB*20/100;
8 elseif ND>10
9 DES=SB*15/100;
10 elseif ND>5
11 DES=SB*10/100;
12 else
13 DES=0;
14 end
15 PF=SB-DES;
16 fprintf('Subtotal a pagar % .2f \n',SB)
17 fprintf('Descuento % .2f \n',DES)
18 fprintf('Total a pagar % .2f \n',PF)
```



```
Command Window
Ingrese el numero de dias: 12
Ingrese el precio de la habitación: 25
Subtotal a pagar 300.00
Descuento 45.00
Total a pagar 255.00
fx >>
```

```
Command Window
Ingrese el numero de dias: 4
Ingrese el precio de la habitación: 25
Subtotal a pagar 100.00
Descuento 0.00
Total a pagar 100.00
fx >>
```

Figura 2.25: Salida ejemplo 12 estructuras de bifurcación

13. Un local de eventos ofrece platos a la carta sus tarifas son las siguientes: el costo del plato por persona es 9.50, pero si el número de personas es mayor a 200 y menor o igual a 300 el costo es de 8.50. Para más de 300 personas el costo es de 7.50. Se requiere un script que ayude a determinar el presupuesto que se debe presentar a los clientes que desean realizar un evento.

```
1 NP=input('Ingrese el numero de personas: ');
2 if NP>300
3 VP=NP*7.50;
4 elseif NP>200
5 VP=NP*8.50;
6 else
7 VP=NP*9.50;
8 end
9 fprintf('Por %d personas debe cancelar en el evento % .2f \n',NP,VP)
```

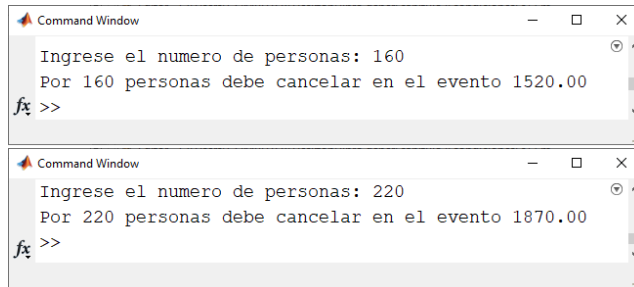


Figura 2.26: Salida ejemplo 13 estructuras de bifurcación

### 2.5.3. Bifurcación compuesta

Es representado por la bifurcación simple o anidada con la diferencia de que en vez de utilizar una sola pregunta en el bloque de condición puede utilizar 2 o más preguntas en un solo bloque, unidas estas preguntas por medio de los operadores lógicos AND y OR, los que tienen como característica trabajar con 2 o más preguntas, cada una de estas preguntas pudiendo ser verdadera o falso.

14. Realizar un script que indique si un número X leído cumple o no las siguientes condiciones, que sea entero y positivo.

```

1 clear
2 clc
3 X=input('Ingrese un numero: ');
4 if X>0 & X==floor(X)
5 disp('Si cumple')
6 else
7 disp('no cumple')
8 end

```

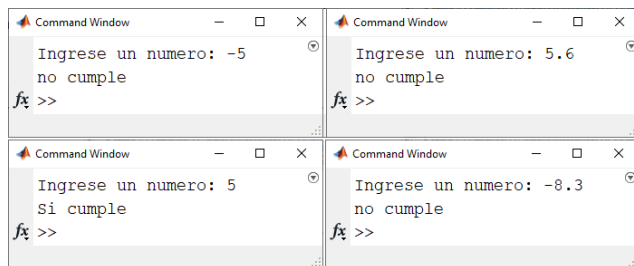


Figura 2.27: Salida ejemplo 14 estructuras de bifurcación

15. Determinar el descuento y el valor que debe pagar por una compra realizada por un cliente, de acuerdo a lo siguiente: 0 - 100 no hay descuento 101 - 200 1% 201 - 500 2% 501 a más 2.5%

```

1 clear
2 clc
3 C=input('Ingrese el valor de la compra: ');
4 if C>=0 & C<=100
5 D=0;
6 elseif C>=101 & C<=200
7 D=C*1/100;
8 elseif C>=201 & C<=500
9 D=C*2/100;
10 else

```

```

11 D=C*2.5/100;
12 end
13 VP=C-D;
14 fprintf('Descuento %.2f Valor a pagar %.2f \n',D,VP)

```

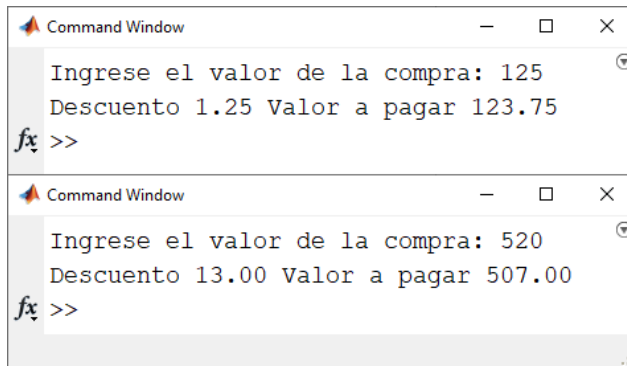


Figura 2.28: Salida ejemplo 15 estructuras de bifurcación

16. Calcular la utilidad que se le debe pagar a un trabajador anualmente por sus años de servicio, si se calcula en base al salario mensual. Esto se basa en la antigüedad que puede tener el empleado de acuerdo a lo siguiente: De 0 a 1 año recibe el 5% del salario de 1 a 5 años recibe el 7% de 5 años a 10 años recibe el 9% y si tiene mas de 10 años de antigüedad recibe el 11%.

```

1 clear
2 clc
3 SM=input('Ingrese el salario mensual: ');
4 A=input('Ingrese el numero de años de servicio: ');
5 if A>=0 & A<=1
6 U=SM*5/100;
7 elseif A>1 & A<=5
8 U=SM*7/100;
9 elseif A>5 & A<=10
10 U=SM*9/100;
11 else
12 U=SM*11/100;
13 end
14 UT=U*12*A;
15 fprintf('La utilidad a pagar por %d años de servicio es %.2f \n',A,UT)

```

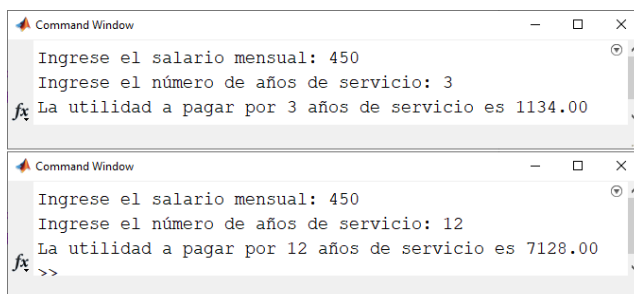


Figura 2.29: Salida ejemplo 16 estructuras de bifurcación

## 2.5.4. Bifurcación Múltiple

También conocida como estructura tipo CASE o MENU, tiene como características que por medio de una condición que se ejecuta automáticamente permite seleccionar uno de varios caminos

posibles, cada uno de estos contiene un proceso diferente.

Así también como otra característica se acostumbra por lo general antes de cualquier proceso desplazar en un bloque el menú o las opciones. Tanto el ingreso de datos como la salida de resultados puede darse uno solo para todo el proceso o una diferente para cada uno.

17. Realizar un script que calcule individualmente cada uno de los parámetros del movimiento rectilíneo uniforme.

```
1 clc
2 disp('1.- Velocidad')
3 disp('2.- Espacio')
4 disp('3.- Tiempo')
5 X=input('Selecciones una opcion: ');
6 switch X
7 case 1
8 E=input('Ingrese el valor del espacio: ');
9 T=input('Ingrese el valor del tiempo: ');
10 V=E/T;
11 fprintf('El valor de la velocidad es %.2f \n',V)
12 case 2
13 V=input('Ingrese el valor de la velocidad: ');
14 T=input('Ingrese el valor del tiempo: ');
15 E=V*T;
16 fprintf('El valor del espacio es %.2f \n',E)
17 case 3
18 E=input('Ingrese el valor del espacio: ');
19 V=input('Ingrese el valor de la velocidad: ');
20 T=E/V;
21 fprintf('El valor del tiempo es %.2f \n',T)
22 otherwise
23 disp('Opcion incorrecta')
24 end
```

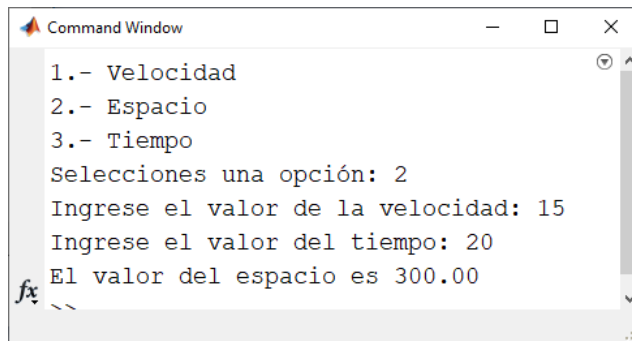


Figura 2.30: Salida ejemplo 17 estructuras de bifurcación

18. Determinar la tarifa aérea de 3 diferentes ciudades.

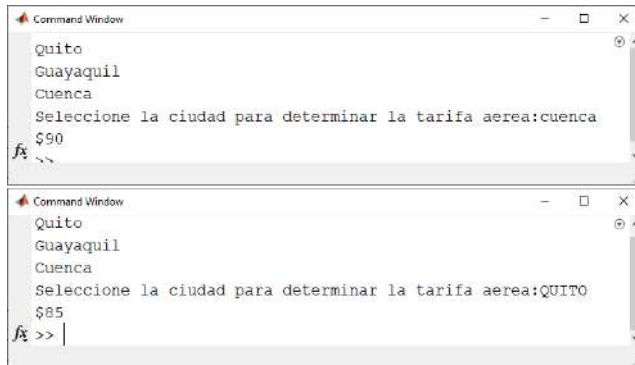
```
1 clear
2 clc
3 disp('Quito')
4 disp('Guayaquil')
5 disp('Cuenca')
6 c=input('Seleccione la ciudad para determinar la tarifa aerea:', 's');
7 switch c
8 case {'Quito', 'quito', 'QUITO'}
9 disp('$85')
10 case {'Guayaquil', 'guayaquil', 'GUAYAQUIL'}
11 disp('$95')
12 case {'Cuenca', 'CUENCA', 'cuenca'}
13 disp('$90')
14 otherwise
```



```

15 disp('No existe la opcion')
16 end

```



**Figura 2.31:** Salida ejemplo 18 estructuras de bifurcación

Cuando las opciones son cadenas de caracteres se debe tomar en cuenta las posibles opciones que el usuario puede ingresar entre ellas se puede mencionar todo en mayúsculas, todo en minúsculas, la primera mayúscula y lo demás minúscula.

Si el texto no se ingresa como una de las posibilidades colocadas en las opciones no la toma como verdadera. Una posible solución es utilizar la función menú de Matlab.

**Función menú:** se usa con frecuencia en conjunto con una estructura switch/case. Esta función hace que aparezca un recuadro de menú en la pantalla, con una serie de botones definidos por el programador.

Sintaxis:

```
variable= menu('Mensaje al usuario', 'texto botón 1','texto botón 2',etc.)
```

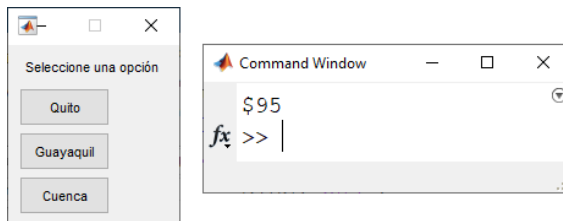
**Ejemplos:**

19. Resolver el ejercicio 17 utilizando la función menú.

```

1 clc
2 c=menu('Seleccione una opcion','Quito','Guayaquil','Cuenca');
3 switch c
4 case 1
5 disp('$85')
6 case 2
7 disp('$95')
8 case 3
9 disp('$90')
10 end

```



**Figura 2.32:** Salida ejemplo 19 estructuras de bifurcación

20. Calcular el área de las siguientes figuras geométricas: cuadrado, rectángulo, triángulo.

```

1 clc
2 disp('Seleccione una opcion')
3 x=menu('Seleccione una opcion','Cuadrado','Rectangulo','Triangulo');
4 switch x
5 case 1
6 l=input('Ingrese el lado: ');
7 a=l*l;
8 fprintf('El area del cuadrado %.2f\n',a)
9 case 2
10 b=input('Ingrese la base: ');
11 h=input('Ingrese la altura: ');
12 a=b*h;
13 fprintf('El area del rectangulo %.2f \n',a)
14 case 3
15 b=input('Ingrese la base: ');
16 h=input('Ingrese la altura: ');
17 a=b*h/2;
18 fprintf('El area del triangulo %.2f \n',a)
19 otherwise
20 disp('No existe esa opcion');
21 end

```

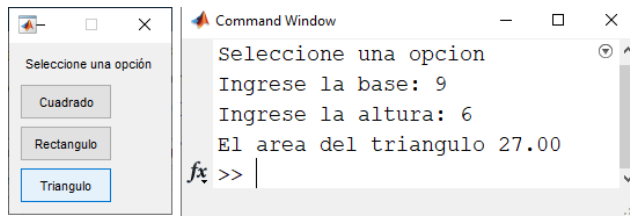


Figura 2.33: Salida ejemplo 20 estructuras de bifurcación

### 2.5.5. Ejercicios propuestos estructuras de control

1. Realizar un script para determinar cuánto se debe pagar por x cantidad de lápices considerando que si son 100 o más el costo es de 0.85 centavos; de lo contrario es de 0.90 centavos.
2. Calcular el área del terreno leyendo el largo por el ancho, solo que el si el terreno es mayor que 400 m<sup>2</sup> y menor que 500 m<sup>2</sup> se hace un descuento de 10% en el precio por m<sup>2</sup>. Si es mayor igual a 500 y menor a 999 metros<sup>2</sup> se hace un descuento de 17%, a partir de 1000 metros<sup>2</sup>, el descuento es de 25%. Obtener el valor a cancelar en la venta de este terreno.
3. Desarrolle un script que calcule y muestre el monto que debe pagar un usuario por concepto de consumo de luz eléctrica y servicio de aseo urbano. Dicho monto se calcula multiplicando la diferencia de la lectura anterior y la lectura actual por el costo de cada Kilovatio hora, según la siguiente escala: 0 - 100 \$0.20 cada Kwh 101 - 300 \$0.30 cada Kwh 301 - 500 \$0.4 cada Kwh 501 - en adelante \$0.50 cada Khw El aseo urbano es el 0.1% del valor del consumo
4. El secretario de salud ha decidido otorgar un bono por desempeño a todos los médicos con base en la puntuación siguiente:
  - 0 - 100 Puntos recibe 1 salario
  - 101 - 150 Puntos recibe 2 salarios mínimos
  - 151 Puntos en adelante recibe 3 salarios mínimos
 Realice un script que determine el valor q recibirá el medico como bono.
5. En una empresa que contrata solo empleados nacionales todos ganan un mismo sueldo con la diferencia de que ganan un porcentaje más dependiendo de la región de donde provienen con las

siguientes condiciones.

Proviene del oriente gana un 40 % más el sueldo básico  
Región insular 50 %  
Costa 30 %  
Sierra 20 %

## 2.6. Estructuras de repetición

Los bucles permiten repetir un número determinado de veces un conjunto de sentencia o instrucciones muchas veces.

Se tiene las siguientes estructuras de repetición:

### 2.6.1. Estructura while - do

Esta estructura permite repetir un conjunto de instrucciones mientras la condición sea verdadera.

Sintaxis:

```
while (condición)
 instrucciones (se ejecutan cuando la condición es verdadera)
end
```

**Ejemplos:**

1. Realizar un script de imprima n veces la frase 'PROGRAMACION CON MATLAB'.

```
1 clc
2 N=input('Ingrese el numero de veces que se va a imprimir: ');
3 I=1;
4 while I<=N
5 disp('PROGRAMACION CON MATLAB')
6 N=N-1;
7 end
```

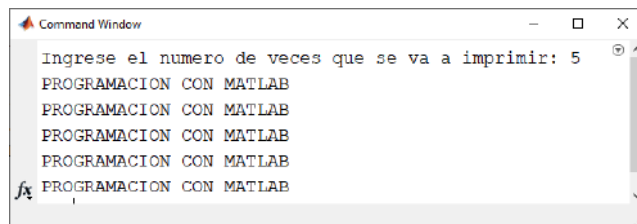


Figura 2.34: Salida ejemplo 1 estructuras de repetición

2. Realizar un script que imprima los n primeros números impares.

```
1 clear
2 clc
3 n=input('Ingrese la cantidad de numeros impares q desea generar: ');
4 p=1; r=1;
5 while p<=n
6 disp(r)
7 p=p+1;
```

```

8 r=r+2;
9 end

```

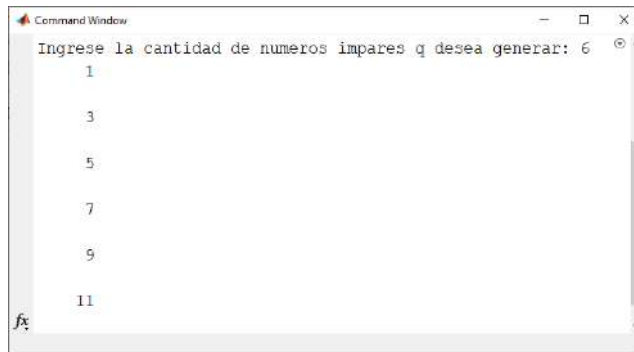


Figura 2.35: Salida ejemplo 2 estructuras de repetición

3. Realizar un script que imprima los n primeros números pares y su sumatoria total.

```

1 clear
2 clc
3 s=0;
4 n=input('Ingrese la cantidad de numeros pares q desea generar: ');
5 p=1; r=0;
6 while p<=n
7 disp(r)
8 p=p+1;
9 s=s+r;
10 r=r+2;
11 end
12 fprintf('La sumatoira es %d \n',s)

```

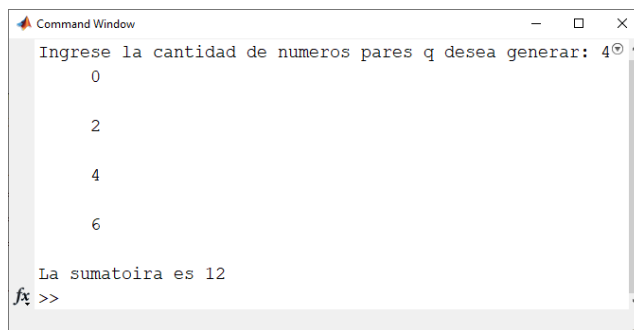


Figura 2.36: Salida ejemplo 3 estructuras de repetición

4. Realizar un script que imprima los n primeros números de la siguiente serie 1,5,9,13,17,21,25. ...

```

1 clear
2 clc
3 n=input('Ingrese la cantidad de numeros pares q desea generar: ');
4 p=1; r=1;
5 while p<=n
6 disp(r)
7 p=p+1;
8 r=r+4;
9 end

```

```

Command Window
Ingrese la cantidad de numeros pares q desea generar: 5
1
5
9
13
17

```

Figura 2.37: Salida ejemplo 4 estructuras de repetición

5. Realizar un diagrama de flujo que imprima los n primeros números de la siguiente serie  $\frac{4}{5}, \frac{7}{9}, \frac{10}{13}, \frac{13}{17}, \dots$

```

1 clear
2 clc
3 n=input('Ingrese la cantidad de numeros pares q desea generar: ');
4 p=1; num=4; den=5;
5 while p<=n
6 fprintf('%d/%d \n',num,den)
7 p=p+1;
8 num=num+3;
9 den=den+4;
10 end

```

```

Command Window
Ingrese la cantidad de numeros pares q desea generar: 4
4/5
7/9
10/13
13/17
>>

```

Figura 2.38: Salida ejemplo 5 estructuras de repetición

6. Realice un script que genere la tabla de multiplicar hasta el 12 de un número ingresado por el usuario.

```

1 clc
2 X=input('Ingrese el numero del cual desea generar la tabla de multiplicar:');
3 I=1;
4 while I<=12
5 R=I*X;
6 fprintf('%d * %d = %d \n',I,X,R)
7 I=I+1;
8 end

```

```

Command Window
Ingrese el numero del cual desea generar la tabla de multiplicar:5
1 * 5 = 5
2 * 5 = 10
3 * 5 = 15
4 * 5 = 20
5 * 5 = 25
6 * 5 = 30
7 * 5 = 35
8 * 5 = 40
9 * 5 = 45
10 * 5 = 50
11 * 5 = 55
12 * 5 = 60
>>

```

Figura 2.39: Salida ejemplo 6 estructuras de repetición

7. Realizar un script que imprima cada uno de los términos de la siguiente serie.  $S = 2, -4, -8, 10, -12, \dots$

```

1 N=input('Ingrese la cantidad de terminos de la serie a imprimir: ');
2 A=2;
3 B=-4;
4 I=1;
5 while I<=N
6 if rem(I,2)==0
7 disp(B)
8 B=B-4;
9 else
10 disp(A)
11 A=A+4;
12 end
13 I=I+1;
14 end

```

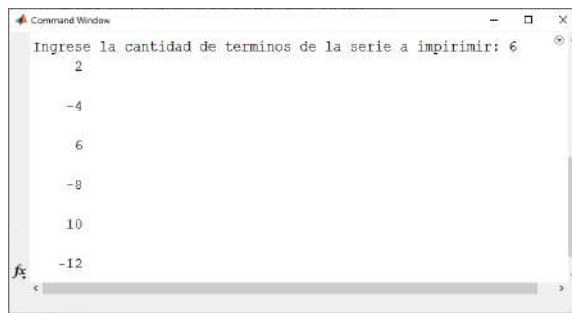


Figura 2.40: Salida ejemplo 7 estructuras de repetición

8. Realizar un script que permita ingresar n números e imprima todos los números ingresados menores e iguales a 5.

```

1 N=input('Ingrese la cantidad de terminos de la serie a imprimir: ');
2 A=2;
3 B=-4;
4 I=1;
5 while I<=N
6 X= input('Ingrese un numero: ');
7 if X<=5
8 disp(X)
9 end
10 I=I+1;
11 end

```

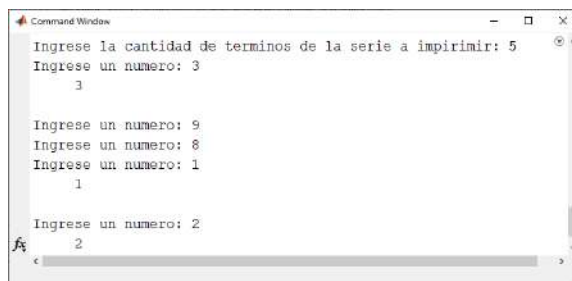


Figura 2.41: Salida ejemplo 8 estructuras de repetición

9. Realizar un script que determine el factorial de un número.

```

1 clear
2 clc
3 n=input('Ingrese un numero: ');
4 fact=1;j=1;
5 while j<=n
6 fact=fact*j;
7 j=j+1;
8 end
9 disp(fact)

```

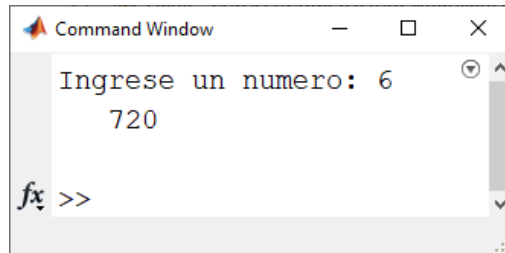


Figura 2.42: Salida ejemplo 9 estructuras de repetición

10. Realizar un script que lea una secuencia de números que finaliza en cero y muestre el cuadrado de cada uno de los números ingresados excepto el último.

```

1 clear
2 clc
3 x=input('Ingrese un numero: ');
4 while x<=0
5 c=x^2;
6 disp(c)
7 x=input('Ingrese un numero: ');
8 end

```

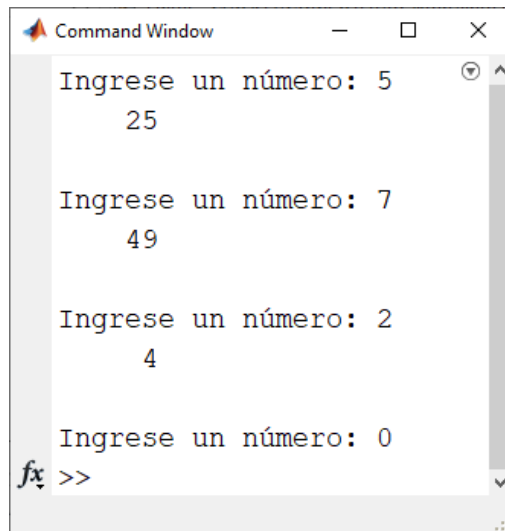


Figura 2.43: Salida ejemplo 10 estructuras de repetición

## 2.6.2. Estructura for

La estructura for es un bucle para repetir un número determinado de veces.

```
for indice = valores
 instrucciones
end
```

valores puede tener los siguientes formatos:

**indice=valor\_inicial:valor\_final** incrementa la variable índice desde un valor inicial a un valor final en 1 y repite la ejecución de las instrucciones hasta que la variable índice sea mayor que el valor final.

**indice=valor\_inicial:incremento:valor\_final** incrementa la variable índice en el valor incremento en cada iteración o disminuye el índice cuando el incremento es negativo.

**indice=vector** crea un vector columna, índice toma valores del arreglo y repite las instrucciones una vez por cada elemento del arreglo en cada iteración.

El bucle se ejecuta un máximo de n veces, donde n es el número de columnas del arreglo. La entrada vector puede ser de cualquier tipo de datos, incluido un vector de caracteres, un arreglo de celdas o una estructura.

### *Ejemplo:*

```
for v = 10:-2:0
 disp(v)
end
```

```
10
 8
 6
 4
 2
 0
```

```
for v = [2 9 12 24]
 disp(v)
end
```

```
2
 9
12
24
```

11. Realizar un script que imprima n veces la palabra "PROGRAMACION CON MATLAB"

```
1 clear
2 clc
3 N=input('Ingrese el numero de veces a imprimir: ');
4 for K=1:N
5 disp('PROGRAMACION CON MATLAB')
6 end
```



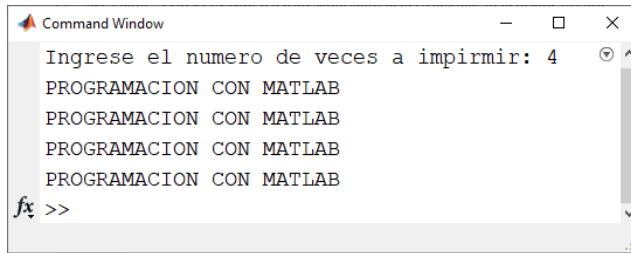


Figura 2.44: Salida ejemplo 11 estructuras de repetición

12. Realizar un script para generar e imprimir los n primeros numeros de la siguiente serie: 1, 1, 2, 2, 4, 3, 8, 4, 16, 5,....

```

1 clear
2 clc
3 N=input('Ingrese la cantidad de terminos de la serie: ');
4 P=1;
5 R=1;
6 for K=1:N
7 if mod(K,2)==0
8 disp(R)
9 R=R+1;
10 else
11 disp(P)
12 P=P*2;
13 end
14 end

```

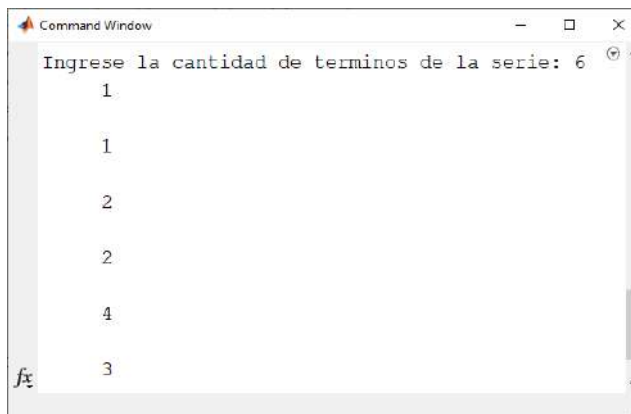


Figura 2.45: Salida ejemplo 12 estructuras de repetición

13. Realizar un script que lea n números y determine cuántos de ellos son positivos y cuantos son negativos

```

1 clear
2 clc
3 N=input('INGRESE LA CANTIDAD DE NUMEROS: ');
4 POS=0;NEG=0;
5 for K=1:N
6 R=input('INGRESE EL NUMERO: ');
7 if R>0
8 POS=POS+1;
9 else
10 NEG=NEG+1;

```

```

11 end
12 end
13 fprintf('EXISTEN %d NUMEROS POSITIVOS \n',POS)
14 fprintf('EXISTEN %d NUMEROS NEGATIVOS \n',NEG)

```

```

Command Window
INGRESE LA CANTIDAD DE NUMEROS: 5
INGRESE EL NUMERO: 8
INGRESE EL NUMERO: -5
INGRESE EL NUMERO: -9
INGRESE EL NUMERO: 3
INGRESE EL NUMERO: 7
EXISTEN 3 NUMEROS POSITIVOS
fx EXISTEN 2 NUMEROS NEGATIVOS

```

Figura 2.46: Salida ejemplo 13 estructuras de repetición

14. Realizar un script que lea 2 números enteros positivos, y obtenga el resultado de multiplicar dichos valores, realizando procesos de sumas solamente.

```

1 X=input('Ingrese un numero: ');
2 Y=input('Ingrese un numero: ');
3 if X>0 & Y>0 & X==floor(X) & Y==floor(Y)
4 S=0;
5 for I=1:Y
6 S=S+X;
7 end
8 disp(S)
9 else
10 disp('No son numeros enteros y positivos')
11 end

```

```

Command Window
Ingrese un numero: -5
Ingrese un numero: 4
No son numeros enteros y positivos
>> r14
Ingrese un numero: 4
Ingrese un numero: 5
fx 20

```

Figura 2.47: Salida ejemplo 14 estructuras de repetición

15. Realizar un script que permita leer 8 números y determine el número mayor de ellos. Se desea que se imprima de la siguiente manera:

```

EL NUMERO 1 INGRESADO ES: 6
EL NUMERO 2 INGRESADO ES: 7
EL NUMERO 3 INGRESADO ES: 5
EL NUMERO 4 INGRESADO ES: 1

```

EL NUMERO 5 INGRESADO ES: 9  
 EL NUMERO 6 INGRESADO ES: 3  
 EL NUMERO 7 INGRESADO ES: 6  
 EL NUMERO 8 INGRESADO ES: 2  
 EL NUMERO MAYOR ES: 9

```

1 M=0;
2 for I=1:8
3 X=input('Ingrese un numero: ');
4 fprintf('El numero %d ingresado es %d \n',I,X)
5 if X>M
6 M=X;
7 end
8 end
9 fprintf('EL numero mayor es: %d \n',M)

```

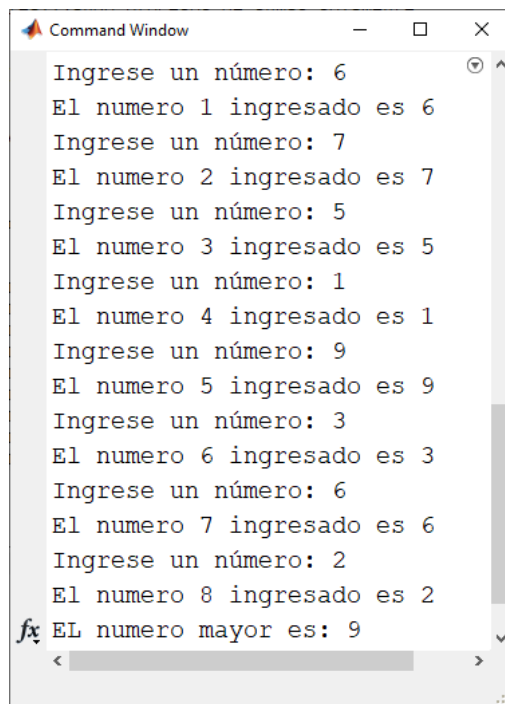


Figura 2.48: Salida ejemplo 15 estructuras de repetición

16. Un lugar de comida rápida ofrece hamburguesas simples (S), dobles (D) y triples (T), las cuales tienen un costo de \$1.50, \$2.50 y \$3.50 respectivamente. Los clientes adquieren N hamburguesas, las cuales pueden ser de diferente tipo, realice un script para determinar cuánto deben pagar.

```

1 clear
2 clc
3 N=input('Ingrese la cantidad de hamburguesas que desea: ');
4 SE=0;DO=0;TR=0;
5 for K=1:N
6 TIPO=menu('Ingrese el tipo de hamburguesa que desea: ...
7 ','Sencilla','Doble','Triple');
8 switch TIPO
9 case 1
10 SE=SE+1;

```

```

11 DO=DO+1;
12 case 3
13 TR=TR+1;
14 end
15 end
16 TP=SE*1.50+DO*2.5+TR*3.50;
17 fprintf('Total a pagar %.2f por %d hamburguesas \n',TP,N)

```

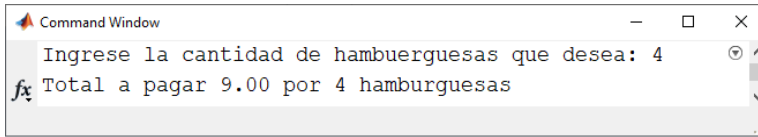


Figura 2.49: Salida ejemplo 16 estructuras de repetición

17. Realizar un script que lea  $n$  números y muestre la tabla de multiplicar del 1 al 9 de los  $n$  números solicitados.

```

1 clear
2 clc
3 N=input('Ingrese la cantidad de numeros para generar las tablas de multiplicar: ');
4 for P=1:N
5 X=input('Ingrese el numero para generar la tabla: ');
6 for K=1:9
7 C=K*X;
8 fprintf('%d * %d = %d \n',K,X,C)
9 end
10 end

```

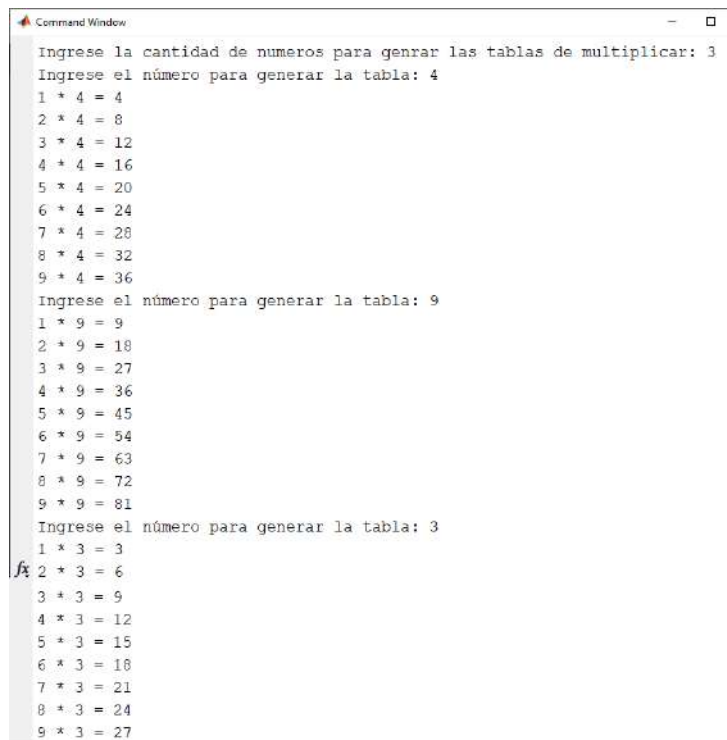


Figura 2.50: Salida ejemplo 17 estructuras de repetición

18. Una compañía fabrica focos de colores (verdes, blancos y rojos). Se desea contabilizar, de un lote de N focos, el número de focos de cada color que hay en existencia y de que color hay la mayor cantidad.

```

1 clear
2 clc
3 N=input('Ingrese la cantidad de focos: ');
4 RO=0;VE=0;BL=0;
5 for P=1:N
6 C=menu('Seleccione el color del foco: ','ROJO','VERDE','BLANCO');
7 switch C
8 case 1
9 RO=RO+1;
10 case 2
11 VE=VE+1;
12 case 3
13 BL=BL+1;
14 end
15 end
16
17 fprintf('Hay %d focos rojos \n',RO)
18 fprintf('Hay %d focos verdes \n',VE)
19 fprintf('Hay %d focos blancos \n',BL)
20
21 if RO>BL & RO>VE
22 fprintf('La mayor cantidad de focos son ROJOS con %d \n',RO)
23 elseif BL>RO & BL>VE
24 fprintf('La mayor cantidad de focos son BLANCOS con %d \n',BL)
25 else
26 fprintf('La mayor cantidad de focos son VERDES con %d \n',VE)
27 end

```

```

Command Window
Ingrese la cantidad de focos: 8
Hay 2 focos rojos
Hay 4 focos verdes
Hay 2 focos blancos
fx La mayor cantidad de focos son VERDES con 4

```

Figura 2.51: Salida ejemplo 18 estructuras de repetición

19. Realizar un script que lea un número de 5 cifras y determine cuantos dígitos del número ingresado son pares y cuantos números impares.

```

1 clear
2 clc
3 N=input('Ingrese un numero: ');
4 if N>=10000 & N<100000
5 X=N;
6 PAR=0;
7 IMPAR=0;
8 for P=1:5
9 C=mod(N,10);
10 N=floor(N/10);
11 if mod(C,2)==0
12 PAR=PAR+1;
13 else
14 IMPAR=IMPAR+1;
15 end
16 end
17 fprintf('En el numero %d existe %d pares y %d impares \n',X,PAR,IMPAR)
18 else
19 disp('No tien 5 cifras')
20 end

```

```

Command Window
Ingrese un número: 76589
En el numero 76589 existe 2 pares y 3 impares

```

Figura 2.52: Salida ejemplo 19 estructuras de repetición

20. Un empleado en una tienda realiza  $N$  ventas durante el día, se requiere saber cuántas de ellas fueron mayores a \$1000, cuántas fueron mayores a \$500 pero menores o iguales a \$1000, y cuántas fueron menores o iguales a \$500. Además, se requiere saber el monto de lo vendido en cada categoría y de forma global.

```

1 clear
2 clc
3 N=input('Ingrese el numero de ventas: ');
4 C1=0;C2=0;C3=0;V1=0;V2=0;V3=0;
5 for P=1:N
6 V=input('Ingrese el valor de la venta: ');
7 if V>1000
8 V1=V1+V;
9 C1=C1+1;
10 elseif V>500
11 V2=V2+V;
12 C2=C2+1;
13 else
14 V3=V3+V;
15 C3=C3+1;
16 end
17 end
18 VT=V1+V2+V3;
19 fprintf('Se realizo %d ventas mayores a 1000 con %d \n',C1,V1)
20 fprintf('Se realizo %d ventas mayores a 500 Y menores a 1000 con %d \n',C2,V2)
21 fprintf('Se realizo %d ventas menores a 500 con %d \n',C3,V3)
22 fprintf('En total se vendio %d \n',VT)

```

```

Command Window
Ingrese el numero de ventas: 4
Ingrese el valor de la venta: 520
Ingrese el valor de la venta: 1020
Ingrese el valor de la venta: 120
Ingrese el valor de la venta: 320
Se realizó 1 ventas mayores a 1000 con 1020
Se realizó 1 ventas mayores a 500 Y menores a 1000 con 520
Se realizó 2 ventas menores a 500 con 440
En total se vendio 1980

```

Figura 2.53: Salida ejemplo 20 estructuras de repetición

### 2.6.3. Sentencia break

Termina la ejecución de un bucle for o while. Las instrucciones que están después de la instrucción break no se ejecutan. En los ciclos anidados, break termina solo el bucle en que se produce. Continúa con las instrucciones posteriores a la instrucción end.

**Ejemplo:**

Realizar la sumatoria de una secuencia de números aleatorios hasta que uno de ellos sea mayor que un número ingresado considerado límite superior, enseguida salga del ciclo con la instrucción break.

```

1 clear
2 clc

```

```

3 limite = 0.5;
4 sum = 0;
5 while 1
6 a = rand;
7 disp(a)
8 if a > limite
9 break
10 end
11 sum = sum + a;
12 end
13 fprintf('La sumatoria es %.4f \n',sum)

```

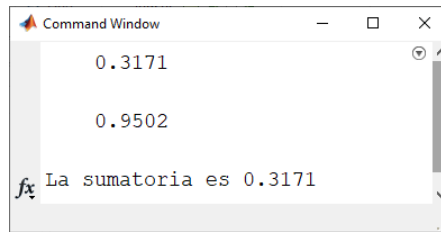


Figura 2.54: Salida ejemplo instrucción break

## 2.6.4. Sentencia continue

La sentencia continue pasa el control a la iteración siguiente en el bucle for o while en el cual aparece ignorando las restantes sentencias en el cuerpo del bucle actual, continue solo se aplica al cuerpo del bucle donde se llama.

### *Ejemplo:*

Realizar un programa que imprima los números del 1 al 20 que no sean múltiplos de 4.

```

1 for n = 1:20
2 if mod(n,4)==0
3 continue
4 end
5 disp(['No es multiple de 4: ' num2str(n)])
6 end

```

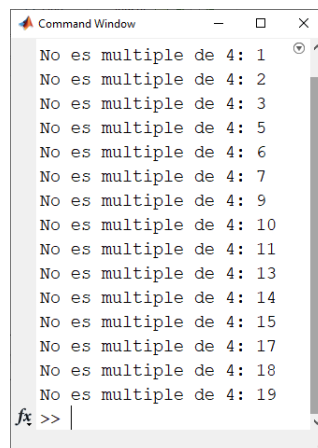


Figura 2.55: Salida ejemplo sentencia continue

### 2.6.5. Ejercicios Propuestos

1. Realizar un script que imprima los  $n$  primeros números de la siguiente serie 1,8,15,22,29,36. . . .
2. Realizar un script que lea 2 valores A y B y obtenga el resultado de elevar A a la potencia B solamente utilizando procesos de multiplicación.
3. Realizar un script para generar e imprimir la siguiente serie 5,2,7,5,9,8,11,11,13,14. . . .
4. Realizar un script para determinar cuánto ahorrará una persona en un año, cada mes deposita diferentes cantidades de dinero; se desea saber cuanto ahorró al terminar el año.
5. Realizar un script que me permita ingresar las calificaciones (0 A 20) de N alumnos y determine el número de aprobados y reprobados. Sabiendo que se aprueba con una calificación  $\geq 16$
6. Realizar un script para determinar, de N números, cuántas son cero, cuántas son menores a cero, y cuántas son mayores a cero.
7. Realizar un script que lea una secuencia de números que finaliza en cero y determine la suma de todos los números ingresados.
8. En una escuela se tiene el registro de las horas que trabaja diariamente un profesor durante la semana (cinco días) se requiere determinar el total de horas trabajadas, y el sueldo que recibirá semanalmente.
9. Realizar un script que permita leer  $n$  números y encontrar el numero mayor de ellos.
10. En una empresa se requiere determinar cuál es la edad promedio de cada uno de los  $n$  departamentos con  $m$  empleados en cada uno y cuál es la edad promedio de toda la empresa. Realizar un script para determinar estos promedios.





# 3

## *Manejo de vectores y matrices con estructuras*





## CAPÍTULO 3

# MANEJO DE VECTORES Y MATRICES CON ESTRUCTURAS

Se caracterizan por el hecho de que con su nombre se hace referencia a un grupo de casillas de memoria. Un dato estructurado puede ser un vector o una matriz.

A diferencia de una variable simple en donde puede tomar varios valores, pero que queda almacenado un valor que es el último ingresado, en un arreglo de igual manera se puede ingresar muchos valores que se almacenan uno a continuación de otro en posiciones no iguales, de esta manera permite ubicar en cualquier momento un dato. Los arreglos se dividen en: UNIDIMENSIONALES (vectores), BIDIMENSIONALES (matrices).

### 3.1. Arreglos unidimensionales o vectores

Un array unidimensional, o lineal, o vector, es un conjunto finito y ordenado de elementos homogéneos.

- Es finito porque tiene un número determinado de elementos.
- Homogéneo porque todos los elementos almacenados van a ser del mismo tipo.
- Ordenado porque vamos a poder acceder a cada elemento de manera independiente, cada elemento puede ser referenciado con la posición en la que se encuentra.

Para identificar un vector se usa al igual que las variables un carácter o un conjunto de caracteres y a continuación encerrados entre paréntesis un subíndice que por lo general es el contador, que recorre desde la primera posición hasta la última posición del vector, notando con esto que para cualquier proceso (lectura, escritura o instrucción se utilizará por lo menos una estructura de repetición para que realice el recorrido).

Nombre(índice)

Por ejemplo:

|      |                                                     |
|------|-----------------------------------------------------|
| V(1) | Se estaría leyendo el vector V en las posiciones 1. |
| V(3) | Se estaría leyendo el vector V en las posiciones 3. |

#### 3.1.1. Lectura de un vector

Para leer un vector y para todo proceso con vectores se necesita una estructura de repetición y además se debe conocer el número de elementos o dimensión del vector.

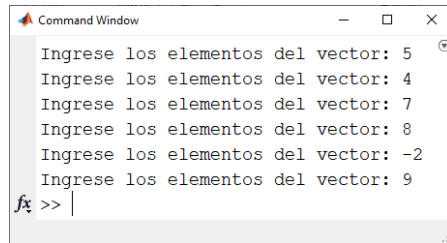
*Ejemplo:*

1. Realizar un script para leer un vector de N elementos.

```

1 clear
2 clc
3 n=input('Ingrese la dimension del vector: ');
4 for i=1:n
5 a(i)=input('Ingrese los elementos del vector: ');
6 end

```



**Figura 3.1:** Salida ejemplo 1 vectores

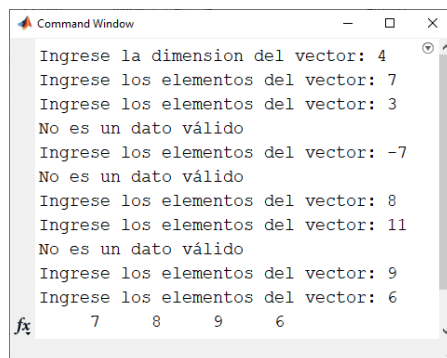
Para realizar cualquier operación con vectores se puede usar la estructura for siempre y cuando que al ingresar los elementos en el vector no se necesite algún tipo de control, es decir controlar que los datos ingresados cumplan alguna característica específica. En ese caso se debe optar por usar la estructura de repetición while que si me permite realizar este tipo de control.

2. Realizar un script para leer un vector de N elementos, en donde todos los elementos ingresados sean positivos, mayores e iguales a 5 y menores e iguales de 10.

```

1 clear
2 clc
3 i=1;
4 n=input('Ingrese la dimension del vector: ');
5 while i<=n
6 c=input('Ingrese los elementos del vector: ');
7 if c>0 & c>=5 & c<=10
8 a(i)=c;
9 i=i+1;
10 else
11 disp('No es un dato v lido')
12 end
13 end
14 disp(a)

```

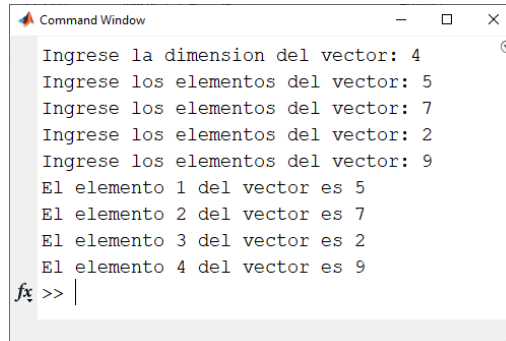


**Figura 3.2:** Salida ejemplo 2 vectores

### 3.1.2. Escritura de un vector

3. Realizar un script para imprimir un vector de N elementos.

```
1 clear
2 clc
3 n=input('Ingrese la dimension del vector: ');
4 for i=1:n
5 a(i)=input('Ingrese los elementos del vector: ');
6 end
7 for i=1:n
8 fprintf('El elemento %d del vector es %d \n',i,a(i))
9 end
```



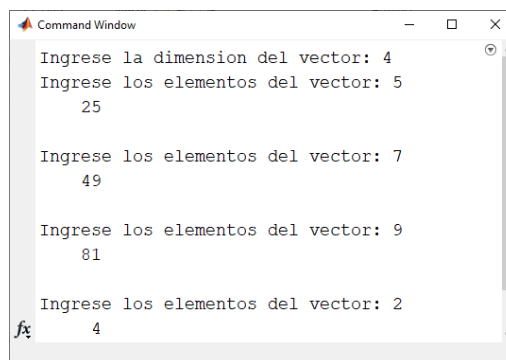
```
Command Window
Ingrese la dimension del vector: 4
Ingrese los elementos del vector: 5
Ingrese los elementos del vector: 7
Ingrese los elementos del vector: 2
Ingrese los elementos del vector: 9
El elemento 1 del vector es 5
El elemento 2 del vector es 7
El elemento 3 del vector es 2
El elemento 4 del vector es 9
fx >> |
```

Figura 3.3: Salida ejemplo 3 vectores

### 3.1.3. Operaciones con vectores

4. Realizar un script que lea un vector de n elementos e imprima cada uno de los elementos del vector elevado al cuadrado.

```
1 clear
2 clc
3 n=input('Ingrese la dimension del vector: ');
4 for i=1:n
5 a(i)=input('Ingrese los elementos del vector: ');
6 c=a(i)^2;
7 disp(c)
8 end
```

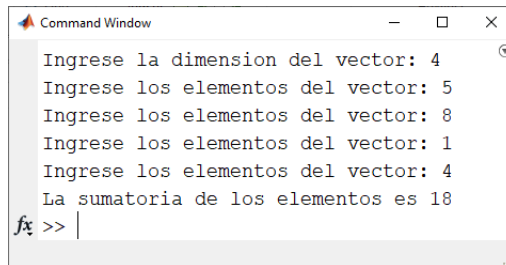


```
Command Window
Ingrese la dimension del vector: 4
Ingrese los elementos del vector: 5
25
Ingrese los elementos del vector: 7
49
Ingrese los elementos del vector: 9
81
Ingrese los elementos del vector: 2
4
fx
```

Figura 3.4: Salida ejemplo 4 vectores

5. Realizar un script que lea un vector de n elementos e imprima la sumatoria de los elementos que se encuentran en el vector.

```
1 clear
2 clc
3 n=input('Ingrese la dimension del vector: ');
4 for i=1:n
5 a(i)=input('Ingrese los elementos del vector: ');
6 end
7 s=0;
8 for i=1:n
9 s=s+a(i);
10 end
11 fprintf('La sumatoria de los elementos es %d \n',s)
```

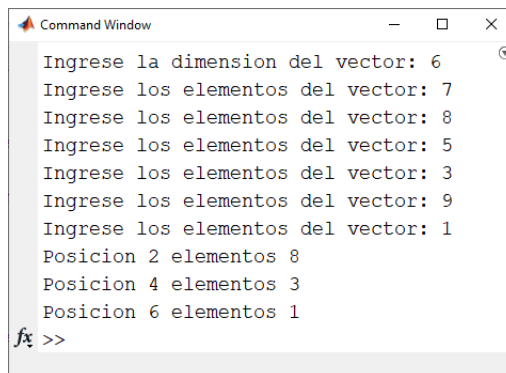


```
Command Window
Ingrese la dimension del vector: 4
Ingrese los elementos del vector: 5
Ingrese los elementos del vector: 8
Ingrese los elementos del vector: 1
Ingrese los elementos del vector: 4
La sumatoria de los elementos es 18
fx >> |
```

Figura 3.5: Salida ejemplo 5 vectores

6. Realizar un script que lea un vector de n elementos e imprima los elementos que se encuentran en las posiciones pares.

```
1 clear
2 clc
3 n=input('Ingrese la dimension del vector: ');
4 for i=1:n
5 a(i)=input('Ingrese los elementos del vector: ');
6 end
7 s=0;
8 for i=1:n
9 if mod(i,2)==0
10 fprintf('Posicion %d elementos %d \n', i,a(i))
11 end
12 end
```



```
Command Window
Ingrese la dimension del vector: 6
Ingrese los elementos del vector: 7
Ingrese los elementos del vector: 8
Ingrese los elementos del vector: 5
Ingrese los elementos del vector: 3
Ingrese los elementos del vector: 9
Ingrese los elementos del vector: 1
Posicion 2 elementos 8
Posicion 4 elementos 3
Posicion 6 elementos 1
fx >>
```

Figura 3.6: Salida ejemplo 6 vectores

7. Realizar un script que lea un vector de n elementos e imprima los elementos impares.

```

1 clear
2 clc
3 n=input('Ingrese la dimension del vector: ');
4 for i=1:n
5 a(i)=input('Ingrese los elementos del vector: ');
6 end
7 s=0;
8 for i=1:n
9 if mod(a(i),2)==1
10 fprintf('Elemento %d \n',a(i))
11 end
12 end

```

```

Command Window
Ingrese la dimension del vector: 6
Ingrese los elementos del vector: 7
Ingrese los elementos del vector: 8
Ingrese los elementos del vector: 4
Ingrese los elementos del vector: 3
Ingrese los elementos del vector: 2
Ingrese los elementos del vector: 9
Elemento 7
Elemento 3
Elemento 9
fx >> |

```

Figura 3.7: Salida ejemplo 7 vectores

8. Realizar un script que lea un vector de n elementos y determine cuantos elementos menores a 5 existen en el vector.

```

1 clear
2 clc
3 n=input('Ingrese la dimension del vector: ');
4 for i=1:n
5 a(i)=input('Ingrese los elementos del vector: ');
6 end
7 c=0;
8 for i=1:n
9 if a(i)<5
10 c=c+1;
11 end
12 end
13 fprintf('En el vector existen %d elementos menores a 5 \n',c)

```

```

Command Window
Ingrese la dimension del vector: 5
Ingrese los elementos del vector: 7
Ingrese los elementos del vector: -2
Ingrese los elementos del vector: 4
Ingrese los elementos del vector: 8
Ingrese los elementos del vector: -1
En el vector existen 3 elementos menores a 5
fx >>

```

Figura 3.8: Salida ejemplo 8 vectores

9. Realizar un script que lea un vector de n elementos y determine cuantas veces se repite un número ingresado por el usuario en el vector.



```

1 clear
2 clc
3 n=input('Ingrese la dimension del vector: ');
4 for i=1:n
5 a(i)=input('Ingrese los elementos del vector: ');
6 end
7 x=input('Ingrese el numero que desea buscar en el vector: ');
8 m=0;
9 for i=1:n
10 if a(i)==x
11 m=m+1;
12 end
13 end
14 fprintf('El numero %d se repite %d en el vector \n',x,m)

```

```

Command Window
Ingrese la dimension del vector: 6
Ingrese los elementos del vector: 45
Ingrese los elementos del vector: 98
Ingrese los elementos del vector: 34
Ingrese los elementos del vector: -56
Ingrese los elementos del vector: 98
Ingrese los elementos del vector: 21
Ingrese el numero que desea buscar en el vector: 98
El numero 98 se repite 2 en el vector
fx >> |

```

Figura 3.9: Salida ejemplo 9 vectores

10. Realizar un script que lea dos vectores de la misma dimensión y almacene en un tercer vector la suma de los dos vectores.

```

1 clear
2 clc
3 n=input('Ingrese la dimension del vector: ');
4 for i=1:n
5 a(i)=input('Ingrese los elementos del primer vector: ');
6 b(i)=input('Ingrese los elementos del segundo vector: ');
7 end
8 for i=1:n
9 c(i)=a(i)+b(i);
10 end
11 for i=1:n
12 fprintf('El elemento %d del vector resultante es %d \n',i,c(i))
13 end

```

```

Command Window
Ingrese la dimension del vector: 4
Ingrese los elementos del primer vector: 5
Ingrese los elementos del segundo vector: 8
Ingrese los elementos del primer vector: 9
Ingrese los elementos del segundo vector: 1
Ingrese los elementos del primer vector: 5
Ingrese los elementos del segundo vector: 2
Ingrese los elementos del primer vector: 7
Ingrese los elementos del segundo vector: 3
El elemento 1 del vector resultante es 13
El elemento 2 del vector resultante es 10
El elemento 3 del vector resultante es 7
El elemento 4 del vector resultante es 10
fx >>

```

Figura 3.10: Salida ejemplo 10 vectores

### 3.1.4. Ejercicios propuestos

1. Realizar un script que lea un vector de n elementos Y determine cuantos elementos son pares y cuantos son impares.
2. Realizar un script que determine cuantos elementos positivos y cuantos elementos negativos y cuantos son cero.
3. Realizar un script que lea un vector de n elementos y cree un nuevo vector con los elementos del vector ingresado que se encuentran en las posiciones impares. Y otro vector con los elementos de las posiciones pares.
4. Realizar un script lea 2 vectores de igual dimensión en los cuales, primer vector se ingresa el nombre de cada uno de los estudiantes del curso y en el segundo una nota por cada estudiante. Se desea encontrar cual es la nota mayor y a quien pertenece.
5. Se tiene almacenadas las calificaciones (entre 5 y 10) de los veinte alumnos del primer semestre, se desea saber el promedio general del grupo.

## 3.2. Arreglos bidimensionales o vectores

En una matriz cada uno de los elementos se referencia por 2 índices, y la representación lógica es una tabla que se subdivide verticalmente en M partes (columnas) y horizontalmente se subdivide en N partes (filas). La posición de un elemento en la matriz se la hace con él numero de la fila seguido con él numero de la columna.

Para identificar a una matriz se utiliza el nombre de la matriz encerrado entre paréntesis el subíndice de la fila y el subíndice de la columna.

$$P(i,j)$$

La dimensión de una matriz viene dada por él numero de filas y por él numero de columnas N\*M elementos.

Cuando el numero de filas es igual al numero de columnas se dice que la matriz es cuadrática. Para realizar cualquier proceso dentro de una matriz se necesita por lo menos dos estructuras de repetición.

### 3.2.1. Lectura de un matriz

Para leer una matriz y para todo proceso con ellas se necesita dos estructuras de repetición, las cuales manejan el índice de las filas y las columnas, para leer se debe conocer el número de elementos o dimensión de la matriz.

#### *Ejemplo:*

1. Realizar un script para leer una matriz de N \* M elementos.

```
1 N=input('Ingrese el numero de filas para la matriz: ');
2 M=input('Ingrese el numero de columnas para la matriz: ');
3 for I=1:N
4 for K=1:M
5 A(I,K)=input('Ingrese los elementos de la matriz: ');
6 end
7 end
8 A
```

```

Command Window
Ingrese el numero de filas para la matriz: 2
Ingrese el numero de columnas para la matriz: 3
Ingrese los elementos de la matriz: 5
Ingrese los elementos de la matriz: 7
Ingrese los elementos de la matriz: 8
Ingrese los elementos de la matriz: 2
Ingrese los elementos de la matriz: 9
Ingrese los elementos de la matriz: 3

A =

 5 7 8
 2 9 3

```

Figura 3.11: Salida ejemplo 1 matrices

### 3.2.2. Escritura de una matriz

2. Realizar un script para imprimir una matriz de  $N \times M$  elementos.

```

1 clear
2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 for I=1:N
6 for K=1:M
7 A(I,K)=input('Ingrese los elementos de la matriz: ');
8 end
9 end
10 for I=1:N
11 for K=1:M
12 fprintf('El elemento %d , %d de la matriz es %d \n',I,K,A(I,K))
13 end
14 end

```

```

Command Window
Ingrese el numero de filas para la matriz: 2
Ingrese el numero de columnas para la matriz: 2
Ingrese los elementos de la matriz: 87
Ingrese los elementos de la matriz: 5
Ingrese los elementos de la matriz: 45
Ingrese los elementos de la matriz: 23
El elemento 1 , 1 de la matriz es 87
El elemento 1 , 2 de la matriz es 5
El elemento 2 , 1 de la matriz es 45
El elemento 2 , 2 de la matriz es 23
>>

```

Figura 3.12: Salida ejemplo 2 matrices

### 3.2.3. Operaciones con matrices

3. Realizar un script que lea una matriz de  $n \times m$  e imprima la sumatoria de los elementos de toda la matriz.

```

1 clear
2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 for i=1:N
6 for j=1:M

```

```

7 P(i,j)=input('Ingrese los elementos de la matriz: ');
8 end
9 end
10 s=0;
11 for i=1:N
12 for j=1:M
13 s=s+P(i,j);
14 end
15 end
16 fprintf('La sumatoria de la matria es %d \n',s)

```

```

Command Window
Ingrese el numero de filas para la matriz: 2
Ingrese el numero de columnas para la matriz: 3
Ingrese los elementos de la matriz: 5
Ingrese los elementos de la matriz: 7
Ingrese los elementos de la matriz: 8
Ingrese los elementos de la matriz: 2
Ingrese los elementos de la matriz: 6
Ingrese los elementos de la matriz: 9
La sumatoria de la matria es 37
fx >>

```

**Figura 3.13:** Salida ejemplo 3 matrices

- Realizar un script que lea una matriz de  $n*m$  e imprima todos los elementos de la matriz excepto la última fila y la primera columna.

```

1 clear
2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 for i=1:N
6 for j=1:M
7 P(i,j)=input('Ingrese los elementos de la matriz: ');
8 end
9 end
10 s=0;
11 for i=1:N-1
12 for j=2:M
13 fprintf('El elemento %d , %d de la matriz es %d \n',i,j,P(i,j))
14 end
15 end

```

```

Command Window
Ingrese el numero de filas para la matriz: 3
Ingrese el numero de columnas para la matriz: 2
Ingrese los elementos de la matriz: 5
Ingrese los elementos de la matriz: 6
Ingrese los elementos de la matriz: 7
Ingrese los elementos de la matriz: 8
Ingrese los elementos de la matriz: 9
Ingrese los elementos de la matriz: 1
El elemento 1 , 2 de la matriz es 6
El elemento 2 , 2 de la matriz es 8
fx >>

```

**Figura 3.14:** Salida ejemplo 4 matrices

- Realizar un script que lea una matriz de  $n*m$  e imprima los números pares que existen en la matriz.

```

1 clear

```

```

2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 for i=1:N
6 for j=1:M
7 P(i,j)=input('Ingrese los elementos de la matriz: ');
8 end
9 end
10 s=0;
11 for i=1:N
12 for j=1:M
13 if mod(P(i,j),2)==0
14 fprintf('El elemento %d , %d es un numero par %d \n',i,j,P(i,j))
15 end
16 end
17 end

```

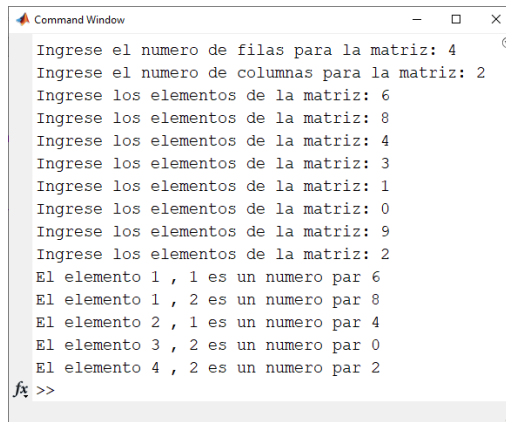


Figura 3.15: Salida ejemplo 5 matrices

6. Realizar un script que lea una matriz de  $n*m$  con valores mayores e iguales a 0 y menores e iguales a 10, e imprima cuantos elementos menores e iguales a 5 existen en la matriz.

```

1 clear
2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 i=1;
6 while i<=N
7 j=1;
8 while j<=M
9 d=input('Ingrese los elementos de la matriz: ');
10 if d>=0 & d<=10
11 P(i,j)=d;
12 j=j+1;
13 else
14 disp('El elemento no esta en el rango!')
15 end
16 end
17 i=i+1;
18 end
19 s=0;
20 for i=1:N
21 for j=1:M
22 if P(i,j)<=5
23 s=s+1;
24 end
25 end
26 end
27 fprintf('En la matriz existen %d elementos menores e iguales a 5 \n',s)

```

```

Command Window
Ingrese el numero de filas para la matriz: 2
Ingrese el numero de columnas para la matriz: 3
Ingrese los elementos de la matriz: 7
Ingrese los elementos de la matriz: -8
El elemento no esta en el rango
Ingrese los elementos de la matriz: 8
Ingrese los elementos de la matriz: 22
El elemento no esta en el rango
Ingrese los elementos de la matriz: 2
Ingrese los elementos de la matriz: 9
Ingrese los elementos de la matriz: 7
Ingrese los elementos de la matriz: 1
en la matraiz existen 2 elementos menores e iguales a 5
fx >> |

```

Figura 3.16: Salida ejemplo 6 matrices

Cuando se debe realizar controles en el ingreso de los datos a una matriz, la estructura for al repetir el proceso una vez por cada elemento de la matriz no es útil, por lo tanto hacemos uso de la estructura while de tal manera que el contador se incremente solamente cuando el valor del elemento ingresado sea correcto, caso contrario no avanza a la siguiente posición.

7. Realizar un script que lea una matriz de  $n*m$  elementos y determine la sumatoria de cada una de las filas.

```

1 clear
2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 for i=1:N
6 for j=1:M
7 B(i,j)=input('Ingrese los elementos de la matriz: ');
8 end
9 end
10 for i=1:N
11 s=0;
12 for j=1:M
13 s=s+B(i,j);
14 end
15 fprintf('La sumatoria de la fila %d es %d \n',i,s)
16 end

```

```

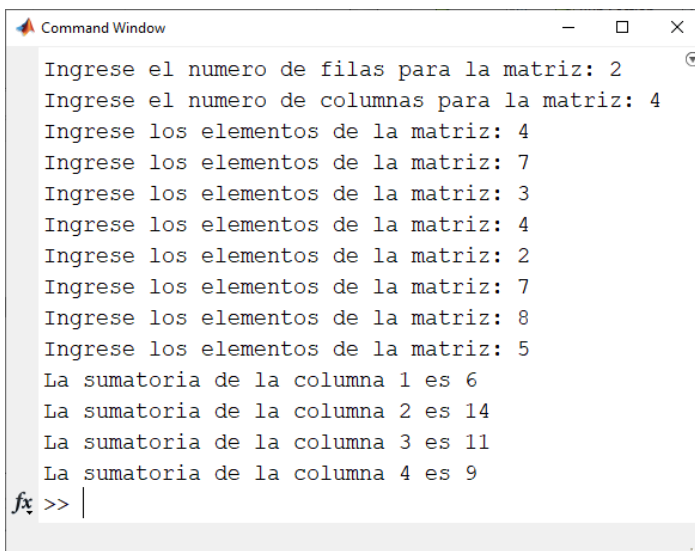
Command Window
Ingrese el numero de filas para la matriz: 3
Ingrese el numero de columnas para la matriz: 4
Ingrese los elementos de la matriz: 6
Ingrese los elementos de la matriz: 7
Ingrese los elementos de la matriz: 8
Ingrese los elementos de la matriz: 4
Ingrese los elementos de la matriz: 3
Ingrese los elementos de la matriz: 5
Ingrese los elementos de la matriz: 1
Ingrese los elementos de la matriz: 2
Ingrese los elementos de la matriz: 3
Ingrese los elementos de la matriz: 6
Ingrese los elementos de la matriz: 7
Ingrese los elementos de la matriz: 3
La sumatoria de la fila 1 es 25
La sumatoria de la fila 2 es 11
La sumatoria de la fila 3 es 19
fx >> |

```

Figura 3.17: Salida ejemplo 7 matrices

8. Realizar un script que lea una matriz de  $n*m$  elementos y determine la sumatoria de cada una de las columnas.

```
1 clear
2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 for i=1:N
6 for j=1:M
7 B(i,j)=input('Ingrese los elementos de la matriz: ');
8 end
9 end
10 for i=1:M
11 s=0;
12 for j=1:N
13 s=s+B(j,i);
14 end
15 fprintf('La sumatoria de la columna %d es %d \n',i,s)
16 end
```



```
Command Window
Ingrese el numero de filas para la matriz: 2
Ingrese el numero de columnas para la matriz: 4
Ingrese los elementos de la matriz: 4
Ingrese los elementos de la matriz: 7
Ingrese los elementos de la matriz: 3
Ingrese los elementos de la matriz: 4
Ingrese los elementos de la matriz: 2
Ingrese los elementos de la matriz: 7
Ingrese los elementos de la matriz: 8
Ingrese los elementos de la matriz: 5
La sumatoria de la columna 1 es 6
La sumatoria de la columna 2 es 14
La sumatoria de la columna 3 es 11
La sumatoria de la columna 4 es 9
fx >> |
```

Figura 3.18: Salida ejemplo 8 matrices

9. Realizar un script que lea una matriz de  $n*m$  y determine el elemento mayor.

```
1 clear
2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 for i=1:N
6 for j=1:M
7 B(i,j)=input('Ingrese los elementos de la matriz: ');
8 end
9 end
10 may=0;
11 for i=1:N
12 for j=1:M
13 if B(i,j)>may
14 may=B(i,j);
15 end
16 end
17 end
18 fprintf('El elemento mayor de la matriz es %d \n',may)
```

```

Command Window
Ingrese el numero de filas para la matriz: 2
Ingrese el numero de columnas para la matriz: 2
Ingrese los elementos de la matriz: 6
Ingrese los elementos de la matriz: 9
Ingrese los elementos de la matriz: 1
Ingrese los elementos de la matriz: 4
El elemento mayor de la matriz es 9
fx >> |

```

Figura 3.19: Salida ejemplo 9 matrices

10. Realizar un script que lea una matriz de  $n \times m$  q encuentre el elemento mayor de cada fila.

```

1 clear
2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 for i=1:N
6 for j=1:M
7 B(i,j)=input('Ingrese los elementos de la matriz: ');
8 end
9 end
10 for i=1:N
11 may=0;
12 for j=1:M
13 if B(i,j)>may
14 may=B(i,j);
15 end
16 end
17 fprintf('El elemento mayor de la fila %d es %d \n',i,may)
18 end

```

```

Command Window
Ingrese el numero de filas para la matriz: 3
Ingrese el numero de columnas para la matriz: 2
Ingrese los elementos de la matriz: 6
Ingrese los elementos de la matriz: 9
Ingrese los elementos de la matriz: 1
Ingrese los elementos de la matriz: 5
Ingrese los elementos de la matriz: 3
Ingrese los elementos de la matriz: 2
El elemento mayor de la fila 1 es 9
El elemento mayor de la fila 2 es 5
El elemento mayor de la fila 3 es 3
fx >>

```

Figura 3.20: Salida ejemplo 10 matrices

### 3.2.4. Ejercicios propuestos

1. Realizar un script que lea una matriz de  $n \times m$  y encuentre el elemento mayor de cada columna.
2. Realizar un script que lea una matriz de  $n \times m$  y determine la sumatoria de los elementos pares que existen en la matriz.



3. Realizar un script que lea una matriz de  $n \times m$  con elementos menores e iguales a 10 e intercambie los elementos mayores a 5 por 0 y los menores e iguales a 5 por 1, imprimir la matriz resultante.
4. Realizar un script que lea una matriz de  $n \times n$  e imprima los elementos que se encuentran en la diagonal principal.
5. Realizar un script que lea dos matrices cuadradas y que determine si la diagonal principal de la primera es igual a la diagonal principal de la segunda matriz.

# 4

## *Funciones definidas por el usuario*





## 4.1. Declaración de funciones

En Matlab la programación se construye en base a funciones, son piezas de código creadas para realizar un proceso específico, o instrucciones que se ejecutan comunmente. Las funciones aceptan argumentos de entrada uno o varios y entrega argumentos de salida uno o varios.

Para almacenar una función se debe tener en cuenta:

- Los nombres de función válidos comienzan con un carácter alfabético.
- Pueden contener letras, números o guiones bajos.
- No debe tener ningún otro carácter especial
- No debe tener espacios en blanco.
- No debe tener palabras reservadas.
- No es aconsejable que el nombre de la función sea muy largo.

Puede guardar la función:

En un archivo de función que solo contenga definiciones de funciones. El nombre del archivo debe coincidir con el nombre de la primera función del archivo.

En un archivo de script que contenga comandos y definiciones de funciones. Las funciones deben estar al final del archivo. Los archivos de script no pueden tener el mismo nombre que una función del archivo.

Las funciones definidas por el usuario se crean en archivos con extensión .m, en la primera línea para crearla función debe tener:

- La palabra function,
- La variable o variables que definan la salida de función.
- El nombre de función
- La variable o variables que se usen como argumentos de entrada.

*Ejemplo:*

```
function salida = mi_funcion(x)
function resultado = suma(a)
```

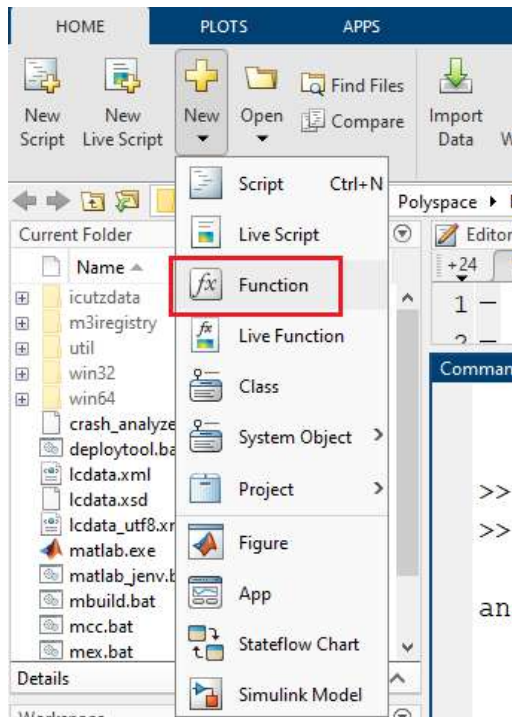


Figura 4.1: Crear archivo para una función

Para crear un archivo de función definida por el usuario debemos ir a Home → *New* → *Function*.

Se nos abre el editor con una plantilla para crear una función como se muestra en el siguiente imagen de lo cual conservaremos lo que se necesite de acuerdo al tipo de función que se vaya a crear.

```

1 function [outputArg1,outputArg2] = untitled(inputArg1,inputArg2)
2 %UNTITLED Summary of this function goes here
3 % Detailed explanation goes here
4 outputArg1 = inputArg1;
5 outputArg2 = inputArg2;
6 end
7
8

```

Figura 4.2: Plantilla de una función

## 4.2. Tipos de funciones

Al igual que Matlab tiene funciones internas en las que se puede tener uno o varios argumentos de entrada y como salida uno o varios argumentos de salida, se puede crear funciones propias con las mismas características.

1. Función con una sola entrada y una sola salida
2. Función con una sola entrada y varias salidas
3. Función con varias entradas y una sola salida

#### 4. Función con varias entradas y varias salidas

##### 4.2.1. Funciones con una sola entrada y una sola salida

Un ejemplo de una función interna de Matlab en la que trabaja con un solo argumento de entrada y devuelve una sola salida es la función trigonométrica coseno.

Para usar la función lo único que se hace es llamarla sea desde la ventana de comando o desde un script y pasarle como argumento un valor para que nos devuelva el coseno de ese número. Dependiendo de como es creada la función el argumento puede ser un valor escalar, un vector o una matriz.

```
>> cos(pi)
ans =
 -1
```

El valor de entrada puede ser directamente un número o una variable.

```
>> x=pi;
>> cos(x)
ans =
 -1
```

De la misma manera si no se especifica la variable para que almacene el resultado de la función, este se almacenará en la variable ans. Si se quiere almacenar el resultado se debe definir un nombre de variable.

```
>> y=cos(x)
y =
 -1
```

Cuando se trabaja con funciones definidas por el usuario el funcionamiento es el mismo, lo primero que se hace es crear el archivo y definir la función con la programación necesaria, luego se la puede utilizar de la misma manera como si fuese una función interna de Matlab.

##### ***Ejemplos:***

1. Crear un función que determine el cuadrado de un número.

```
1 function s = cuadrado(a)
2 %Esta funcion determina el cuadrado de un numero
3 s=a^2;
4 end
```

El argumento de entrada de la función es la variable a mediante la cual ingresan los datos a la función, y el resultado de la función se lo envía a través de la variable o argumento de salida s.

Las funciones no se las ejecuta como los script, a las funciones se las llama o invoca desde la ventana de comandos o script utilizando el nombre de la función que es el mismo nombre del archivo de la siguiente manera:

```
>> cuadrado(5)
ans =
 25
```

El resultado de la función se almacena en la variable ans.

```
>> t=cuadrado(5)
t =
```

El resultado de la función se almacena en la variable t. Las variables de entrada o salida que se utilizan para llamar a la función no necesariamente deben tener el mismo nombre que las que se utiliza dentro de la función como se puede ver en estos ejemplos.

```
b=8;
t=cuadrado(b)
t =
 64
```

La entrada se la envía a la función a través de la variable b y el resultado de la función se almacena en la variable t.

Esta función tal cual esta definida solo acepta como argumentos de entrada valores escalares, si se envía un vector o una matriz nos da error, debido a que para realizar la operación de potencia a un vector se debe usar el operador punto (.) como se explicó el capítulo 1.

```
>> b=[4 7 2 9];
>> t=cuadrado(b)

Error using (line 51)
Incorrect dimensions for raising a matrix to a power. Check that the matrix is square and the power is a scalar.
To perform elementwise matrix powers, use '^.'.

Error in cuadrado (line 3)
s=a.^2;
```

Realizando estos cambios en la función podemos definir una función que acepte argumentos de entrada escalares o matriciales. Sin embargo a pesar de enviar varios datos de entrada(vector) y nos devuelva varios datos de salida(vector) sigue siendo una función con una sola entrada y una sola salida.

```
1 function s = cuadrado1(a)
2 %Esta funcion determina el cuadrado de un numero/s
3 s=a.^2;
4 end

>> b=[4 7 2 9];
>> t=cuadrado1(b)
t =
 16 49 4 81
```

Realizar una función que evalúe un polinomio de tercer orden debe aceptar parámetros de entrada escalares y matriciales.

```
1 function salida=polinomio(x)
2 %Esta funcion evalua un polinomio de cuarto orden
3 salida=5*x.^4+3*x.^3+2*x.^2-6*x-2;

>> x=4;
>> y=polinomio(x)
y =
 1478

>> x=1:5;
>> y=polinomio(x)

y =
 2 98 484 1478 3518
```

Realizar una función llamada promedio que tenga como argumento de entrada un vector, calcule el promedio de los valores y devuelva como salida un solo valor.

```

1 function p = promedio(x)
2
3 %Esta función determina el promedio de un vector
4 p = sum(x(:))/numel(x);
5 end

```

En esta función se utiliza funciones internas de matlab como la función sum que determina la sumatoria de los elementos del vector, y la función numel que devuelve el número de elementos del vector.

```

>> y=[7 5 9 2 4];
>> s=promedio(y)
s =
 5.4000

```

Realizar una función KM que convierta de kilómetros a metros y otra función MK que convierta de metros a kilómetros. Sus funciones deben tener capacidad de aceptar entrada escalar y matricial. Crear un script que permita llamar a las funciones y enviar los datos a ser convertidos. Imprimir una tabla de resultados.

Creamos la función KM

```

1 function m = KM(x)
2
3 %Convierte de kilometros a metros
4 m=x*1000;
5 end

```

Creamos la función MK

```

1 function k = MK(x)
2
3 %Convierte de metros a kilometros
4 k=x/1000;
5 end

```

Creamos el script para llamar a las funciones

```

1 clear
2 clc
3 %definir un valor/es en kilometros
4 kilometros=input('Ingrese el valor/es en kilometros: ');
5
6 %llamar a la funcion
7 metros=KM(kilometros);
8
9 %crear tabla de resultados
10 disp(' kilometros metros')
11 km_met=[kilometros;metros]';
12 disp(km.met)
13
14 %definir un valor/es en metros
15 metros1=input('Ingrese el valor/es en metros: ');
16
17 %llamar a la funcion
18 kilometros1=MK(metros1);
19
20 %tabla de resultados
21 disp(' metros kilometros ')
22 met_km=[metros1;kilometros1]';
23 disp(met_km)

```



```

Command Window
Ingrese el valor/es en kilometros: [2:2:10]
kilometros metros
 2 2000
 4 4000
 6 6000
 8 8000
 10 10000

Ingrese el valor/es en metros: [1000:1000:5000]
metros kilometros
 1000 1
 2000 2
 3000 3
 4000 4
 5000 5

fx >> |

```

Figura 4.3: Ejercicio de funciones

#### 4.2.2. Funciones con varias entradas y una sola salida

Una función interna de matlab en la que se define varias entradas y se obtiene una sola salida, es la función `mod` que devuelve el residuo de una división.

```

>> x=8;y=3;
>> z=mod(x,y)
z =
 2

```

Podemos crear funciones propias de la misma manera en las que se defina varias entradas y se obtenga una sola salida de la función.

**Ejemplos:**

1. Realizar una función que determine la multiplicación de dos vectores suponer que los mismos tienen la misma dimensión, mostrar el vector resultante.

```

1 function salida= multiplica_vector(x,y)
2
3 %multiplicacion de vectores
4 a=x.*y;
5 salida=a;
6 end

```

Llamamos a la función desde la ventana de comandos:

```

>> a=[5 4 8 2];
>> b=[7 3 5 9];
>> c=multiplica_vector(a,b)
c =
 35 12 40 18

```

2. Realizar una función para sumar dos vectores de la misma dimensión usando estructuras de repetición.

```

1 function c=suma_vectores(a,b)
2 dimx=length(a);
3 dimy=length(b);
4 if dimx==dimy
5 for i=1:dimx
6 c(i)=a(i)+b(i);
7 end
8 else
9 c='No tienen la misma dimension';
10 end

```

Llamar a la función:

```

>> a=1:5;
>> b=2:2:10;
>> c=suma_vectores(a,b)

c =
 3 6 9 12 15

>> a=1:5;
>> b=2:2:14;
>> c=suma_vectores(a,b)

c =
 'No tienen la misma dimension'

```

### 4.2.3. Funciones con una sola entrada y varias salidas

La función `size` de Matlab es una función interna en la que se envía una sola entrada y entrega varias salidas. Para invocarla se utiliza entre corchetes nombres de variables para definir las salidas.

```

>> A=[4 6 7;8 9 3];
>> [f,c]=size(A)

f =
 2

c =
 3

```

1. Realizar una función que convierta a kilómetros, metros y centímetros una medida dada en decímetros.

```

1 function [km,m,cm] = longitud(dm)
2 %conversion de decimetros a kilometros, metros y centimetros
3 cm=dm*10;
4 m=dm/10;
5 km=dm/10000;
6 end

```

Para llamar a una función con más de un argumento de salida se debe definir entre corchetes igual número de variables que se definió en la función en el mismo orden. En este ejemplo se utiliza 3 variables.

```

>> [k,m,c]=longitud(34)

k =
 0.0034

m =
 3.4000

c =
 340

```

#### 4.2.4. Funciones con varias entradas y varias salidas

Realizar un función que me permita sumar dos vectores que sean de la misma dimensión y muestre como salida los vectores ingresados y el vector resultante.

```
1 function [a,b,c]=sumar_vectores(x,y)
2 dimx=length(x);
3 dimy=length(y);
4 if dimx==dimy
5 for i=1:dimx
6 c(i)=x(i)+y(i);
7 end
8 a=x;
9 b=y;
10 else
11 c='No tienen la misma dimension';
12 end
```

Llamar a la función:

```
>> x=1:5;
>> y=2:2:10;
>> sumar_vectores(x,y)
```

```
ans =
 1 2 3 4 5
```

Si se invoca o llama a una función de varias salidas con una sola variable no se puede ver a la salida todas las definidas en la función.

```
>> x=1:5;
>> y=2:2:10;
>> c=sumar_vectores(x,y)
```

```
c =
 1 2 3 4 5
```

Correctamente se debe definir cuando se invoque a la función el mismo número de variables que se definio en la función.

```
>> [a,b,c]=sumar_vectores(x,y)
```

```
a =
 1 2 3 4 5
```

```
b =
 2 4 6 8 10
```

```
c =
 3 6 9 12 15
```

#### 4.3. Ejercicios resueltos de funciones

1. Resolver el siguiente ejercicio utilizando funciones definidas por el usuario para ingresar datos, realizar procedimientos e imprimir resultados.

Cierta empresa requiere controlar la existencia de  $n$  productos, los cuales se almacenan en un vector productos, mientras que los pedidos de los clientes de estos productos se almacenan en un vector pedidos. Se requiere generar un tercer vector stock con base en los anteriores que represente lo que se requiere comprar para mantener el stock de inventario, para esto se considera lo siguiente: si los valores correspondientes de los vectores productos y pedidos son iguales se almacena este mismo valor, si el valor de pedidos es mayor que el de productos se almacena el

doble de la diferencia entre pedidos y productos, si se da el caso de que productos es mayor que pedidos, se almacena pedidos, que indica lo que se requiere comprar para mantener el stock de inventario.

Función para lectura de los productos y de los pedidos, utilizando la misma función dos veces.

```
1 function v = leer_vector(n)
2 %Lectura de los datos
3 for i=1:n
4 v(i)=input('Ingrese los elementos del vector: ');
5 end
6 end
```

Función para obtener el stock del inventario.

```
1 function v = stock(n,x,y)
2 for i=1:n
3 if x(i)==y(i)
4 v(i)=x(i);
5 elseif y(i)>x(i)
6 v(i)=2*(y(i)-x(i));
7 else
8 v(i)=y(i);
9 end
10 end
11
12 end
```

Función para imprimir los vectores.

```
1 function v=imprimir_vector(n,x)
2 for i=1:n
3 fprintf('El elemento %d del vector es %d \n',i,x(i))
4 end
```

Script para llamar a las funciones.

```
1 clear
2 clc
3 n=input('Ingrese el numero de productos: ');
4 disp('Existencia de los productos')
5 productos=leer_vector(n);
6
7 disp('Pedidos de los productos')
8 pedidos=leer_vector(n);
9
10 st=stock(n,productos,pedidos);
11
12 disp('Existencia de los productos')
13 imprimir_vector(n,productos)
14
15 disp('Pedidos de los productos')
16 imprimir_vector(n,pedidos)
17
18 disp('Stock del inventario')
19
20 imprimir_vector(n,st)
```

Ejecutamos el script en el cual se va a ir llamando a cada una de las funciones.

Primero se llama a la función para ingresar los elementos del vector, la primera vez que la llama se ingresa los datos de los productos, la segunda que se usa la función leer\_vector se lo hace para leer los datos de los pedidos de los productos.

Una vez ingresado los datos se ejecuta la función para calcular el stock de los productos y finalmente la función para imprimir los vectores resultantes de los procesos.

Ingrese el numero de productos: 6

Existencia de los productos

Ingrese los elementos del vector: 24

Ingrese los elementos del vector: 18

Ingrese los elementos del vector: 15

Ingrese los elementos del vector: 30

Ingrese los elementos del vector: 51

Ingrese los elementos del vector: 12

Pedidos de los productos

Ingrese los elementos del vector: 12

Ingrese los elementos del vector: 9

Ingrese los elementos del vector: 15

Ingrese los elementos del vector: 27

Ingrese los elementos del vector: 65

Ingrese los elementos del vector: 15

Existencia de los productos

El elemento 1 del vector es 24

El elemento 2 del vector es 18

El elemento 3 del vector es 15

El elemento 4 del vector es 30

El elemento 5 del vector es 51

El elemento 6 del vector es 12

Pedidos de los productos

El elemento 1 del vector es 12

El elemento 2 del vector es 9

El elemento 3 del vector es 15

El elemento 4 del vector es 27

El elemento 5 del vector es 65

El elemento 6 del vector es 15

Stock del inventario

El elemento 1 del vector es 12

El elemento 2 del vector es 9

El elemento 3 del vector es 15

El elemento 4 del vector es 27

El elemento 5 del vector es 28

El elemento 6 del vector es 6

2. Resolver el siguiente ejercicio utilizando funciones definidas por el usuario para ingresar datos, realizar procedimientos e imprimir resultados.

Se tienen almacenadas las calificaciones (entre 5 y 10) de  $n$  alumnos del cuarto semestre, se desea saber:

- La frecuencia de las calificaciones (cuantos dieces, cuantos nueves, etc.)
- El promedio general.

Función para lectura de las notas de los alumnos controlando que los datos que se almacenan sean números comprendidos entre 5 y 10

```
1 function notas=leer_notas(n)
2 k=1;
3 while k<=n
4 c=input('Ingrese la calificacion: ');
5 if c>=5 & c<=10
6 notas(k)=c;
7 k=k+1;
8 else
9 disp('Calificacion incorrecta')
10 end
11 end
```

Función para resolver el ítem de la frecuencia de las calificaciones.

```
1 function s=frecuencia(n,notas)
2 ci=0;se=0;si=0;oc=0;nu=0;di=0;
3 for K=1:n
4 if notas(K)==5
5 ci=ci+1;
6 elseif notas(K)==6
7 se=se+1;
8 elseif notas(K)==7
9 si=si+1;
10 elseif notas(K)==8
11 oc=oc+1;
12 elseif notas(K)==9
13 nu=nu+1;
14 else
15 di=di+1;
16 end
17 end
18 fprintf('Existen %d calificaciones igual a 5 \n',ci)
19 fprintf('Existen %d calificaciones igual a 6 \n',se)
20 fprintf('Existen %d calificaciones igual a 7 \n',si)
21 fprintf('Existen %d calificaciones igual a 8 \n',oc)
22 fprintf('Existen %d calificaciones igual a 9 \n',nu)
23 fprintf('Existen %d calificaciones igual a 10 \n',di)
```

Función para resolver el ítem del promedio de las notas.

```
1 function p = promedio(x)
2
3 %Esta funcion determina el promedio de un vector
4 p = sum(x(:))/numel(x);
5 end
```

Script para llamar a las funciones.

```
1 clear
2 clc
3 n=input('Ingrese el numero de alumnos de cuarto semestre: ');
4 Notas=leer_notas(n);
5
6 frecuencia(n,Notas);
7
8 p=promedio_notas(Notas);
9
10 fprintf('El promedio del semestre es %.2f \n',p)
```

Ejecutar el script

Ingrese el numero de alumnos de cuarto semestre: 5  
 Ingrese la calificacion: 34  
 Calificacion incorrecta  
 Ingrese la calificacion: 5  
 Ingrese la calificacion: 7  
 Ingrese la calificacion: 9  
 Ingrese la calificacion: 5  
 Ingrese la calificacion: 12  
 Calificacion incorrecta  
 Ingrese la calificacion: 8

Existen 2 calificaciones igual a 5  
 Existen 0 calificaciones igual a 6  
 Existen 1 calificaciones igual a 7  
 Existen 1 calificaciones igual a 8  
 Existen 1 calificaciones igual a 9  
 Existen 0 calificaciones igual a 10

El promedio del semestre es 6.80

3. Resolver el siguiente ejercicio utilizando funciones definidas por el usuario para ingresar datos, realizar procedimientos e imprimir resultados.

Suponga que se tiene la calificación de n materias de m alumnos que obtuvieron durante el semestre. Esta información se puede almacenar de tal forma que los filas representen las materias, mientras que las columnas representen cada alumnos.

- Obtenga la mayor nota por materia
- Determine los promedios de cada alumno

Función para lectura de las notas de m materia de n alumnos

```

1 function A=leer_matriz(N,M)
2 %ingresa datos a una matriz
3 for I=1:N
4 fprintf('Materia %d \n',I)
5 for K=1:M
6 A(I,K)=input('Ingrese las notas: ');
7 end
8 end

```

Función para obtener mayor nota por materia.

En este caso se esta determinando en otras palabras el número mayor de cada fila.

```

1 function s=mayor_materia(N,M,A)
2 for I=1:N
3 MAY=A(I,1);
4 for K=1:M
5 if A(I,K)>MAY
6 MAY=A(I,K);
7 end
8 end
9 fprintf('La nota mayor de la materia %d es %d \n',I,MAY)
10 end

```

Función para determinar el promedio de cada alumno.

En otras palabras estamos hablando de obtener el promedio de cada columna, para lo cual lo que se hace es sumar las columnas y luego dividir para el numero de filas.

```
1 function s=promedio_alumnos(N,M,A)
2 for I=1:M
3 SUM=0;
4 for K=1:N
5 SUM=SUM+A(K,I);
6 end
7 s=SUM/N;
8 fprintf('El promedio del estudiante %d es %.2f \n',I,s)
9 end
```

Script para llamar a las funciones.

```
1 clear
2 clc
3 N=input('Ingrese el nmero de materias: ');
4 M=input('Ingrese el nmero de alumnos: ');
5 %llamar a la funcion
6 A=leer_matriz(N,M);
7 %obtener el mayor de cada fila
8 mayor_materia(N,M,A)
9 %obtener el promedio de cada columna
10 promedio_alumnos(N,M,A);
```

Ejecutamos el script e ingresamos los datos a la matriz.

Ingrese el número de materias: 2  
Ingrese el número de alumnos: 3

Materia 1  
Ingrese las notas: 12  
Ingrese las notas: 17  
Ingrese las notas: 15

Materia 2  
Ingrese las notas: 18  
Ingrese las notas: 11  
Ingrese las notas: 19

La nota mayor de la materia 1 es 17  
La nota mayor de la materia 2 es 19

El promedio del estudiante 1 es 15.00  
El promedio del estudiante 2 es 14.00  
El promedio del estudiante 3 es 17.00

## 4.4. Ejercicios propuestos

Resolver el siguiente ejercicio utilizando funciones definidas por el usuario para ingresar datos, realizar procedimientos e imprimir resultados.

1. En una escuela los directivos requieren determinar cuál es la edad promedio de cada uno de los  $n$  salones de clases con  $m$  alumnos en cada uno y cuál es la edad promedio de toda la escuela. Realice un programa para determinar estos promedios.



2. Realizar un programa que lea un vector de  $n$  elementos e imprima el elemento mayor considerando que los elementos ingresados pueden ser solo números positivos, solo números negativos o números positivos y negativos.
3. Realizar un programa lea dos vectores de  $n$  elementos y clasifique los elementos de dichos vectores de tal manera que en un vector queden solo elementos pares y en otro solo impares.
4. Realizar un programa que lea una matriz de  $n*m$  determine cuantos elementos menores e iguales a 5 existen en la matriz y la sumatoria de cada una de las filas.
5. Para resolver el ejercicios utilizar una matriz que almacene información de las calificaciones (de 0 a 10) finales de los tres parciales de una materia en un grupo de  $n$  alumnos. Realizar un programa que proporcione la siguiente información.
  - El promedio por alumno
  - El promedio del grupo por parcial

# 5

## *Solución ejercicios propuestos*





**5.1. Solución Ejercicios propuestos Capitulo 1 numeral 1.6**

1. Ejercicio 1

$$\gg g=1+3/4$$

2. Ejercicio 2

$$\gg k=(5+3)/(9-1)$$

3. Ejercicio 3

$$\gg p=(2^3)-(4/(5+3))$$

4. Ejercicio 4

$$\gg j=(4+1/2)*(5+2/3)$$

5. Ejercicio 5

$$\gg r=(9+(6/12))+(7*5^{\hat{2}}(3+2))$$

6. Ejercicio 6

$$\gg a=2(5-3^{\hat{2}}/4)+(-2)^{\hat{-3}}$$

7. Ejercicio 7

$$\gg b=(5-(1/2)^{\hat{2}})/(0.7+1)$$

8. Ejercicio 8

$$\gg c=(1-0.25)^{\hat{1/2}}+(4/81)^{\hat{-1/2}}$$

9. Ejercicio 9

$$\gg d=\text{sqrt}(\text{sqrt}(256)-\text{sqrt}((1/25)^{\hat{-2}}))$$

10. Ejercicio 10

$$\gg f=((5-i)^{\hat{2}}/3i)+(\text{sqrt}(2-i))$$

## 5.2. Solución Ejercicios propuestos capitulo 1 numeral 1.9

### 1. Ejercicio 1

```
>> x=linspace(0,1,80)
```

### 2. Ejercicio 2

```
>> x=[19 0.8 1.51 3.87 10];
>> d=3;
>> p=x(d)
```

### 3. Ejercicio 3

```
>> a=5:4:20
>> b=a.^3
>> c=cumsum(a)
>> d=prod(a)
```

### 4. Ejercicio 4

```
>> p=0:5:295
>> x=20:-3:10;
>> y=30:2:60;
>> r=[p(x) p(y)]

>> s=[4+3i -5i 9-2i 8+4i 5+3i]
>> a=cumsum(s)
>> b=cumprod(s)
>> m=max(s)
>> n=min(s)
```

### 5. Ejercicio 5

```
>> luz=[22 18 26 25 32 17 19 23 21 18 19 20]
>> p=mean(luz)
```

## 5.3. Solución Ejercicios propuestos Capitulo 2

### 1. Ejercicio 1

```
1 clear
2 clc
3 n=input('Ingrese la cantidad de numeros de la serie que desea generar: ');
4 p=1; r=1;
5 while p<=n
6 disp(r)
7 p=p+1;
8 r=r+7;
9 end
```

### 2. Ejercicio 2

```
1 X=input('Ingrese un numero: ');
2 Y=input('Ingrese un numero: ');
3 if X>0 & Y>0 & X==floor(X) & Y==floor(Y)
4 S=1;
```

```

5 for I=1:Y
6 S=S+X;
7 end
8 disp(S)
9 else
10 disp('No son numeros enteros y positivos')
11 end

```

### 3. Ejercicio 3

```

1 clear
2 clc
3 N=input('Ingrese la cantidad de terminos de la serie: ');
4 P=5;
5 R=2;
6 for K=1:N
7 if mod(K,2)==0
8 disp(R)
9 R=R+3;
10 else
11 disp(P)
12 P=P+2;
13 end
14 end

```

### 4. Ejercicio 4

```

1 clear
2 clc
3 P=0;
4 for K=1:12
5 fprintf('Deposito del mes %d \n',K)
6 D=input('Ingrese la cantidad del deposito: ');
7 P=P+D;
8 end
9 fprintf('Al finalizar el anio ahorro %.2f dolares\n',P)

```

### 5. Ejercicio 5

```

1 clear
2 clc
3 N=input('Ingrese el numero de alumnos: ');
4 APRO=0;
5 REPRO=0;
6 K=1;
7 while K<=N
8 C=input('Ingrese la calificacion: ');
9 if C>=0 & C<=20
10 if C>=16
11 APRO=APRO+1;
12 else
13 REPRO=REPRO+1;
14 end
15 K=K+1;
16 else
17 fprintf('Calificacion incorrecta \n')
18 end
19 end
20 fprintf('Estudiantes aprobados %d estudiantes reprobados %d \n',APRO,REPRO)

```

### 6. Ejercicio 6

```

1 clear
2 clc
3 N=input('Ingrese la cantidad de numeros: ');
4 A=0;
5 B=0;

```

```

6 D=0;
7 for K=1:N
8 C=input('Ingrese un numero: ');
9 if C==0
10 A=A+1;
11 elseif C>0
12 B=B+1;
13 else
14 D=D+1;
15 end
16 end
17 fprintf('Numeros igual a cero %d \n',A)
18 fprintf('Numeros mayor o igual a cero %d \n',B)
19 fprintf('Numeros menor o igual a cero %d \n',D)

```

## 7. Ejercicio 7

```

1 clear
2 clc
3 x=input('Ingrese un numero: ');
4 s=0;
5 while x≠0
6 s=s+x;
7 x=input('Ingrese un numero: ');
8 end
9 fprintf('La suma de los numeros ingresados es %d \n',s)

```

## 8. Ejercicio 8

```

1 clear
2 clc
3 S=input('Ingrese el precio por hora que trabaja: ');
4 T=0;
5 for K=1:5
6 fprintf('Dia %d \n',K)
7 H=input('Ingrese el numero de horas que trabajo: ');
8 T=T+H;
9 end
10 VP=T*S;
11 fprintf('El numero de horas trabajadas %d \n',T)
12 fprintf('Sueldo a pagar %d \n',VP)

```

## 9. Ejercicio 9

```

1 clear
2 clc
3 MAY=0;
4 N=input('Ingrese la cantidad de numeros: ');
5 for K=1:N
6 M=input('Ingrese un numero: ');
7 if M>MAY
8 MAY=M;
9 end
10 end
11 fprintf('El numero mayor es %d\n',MAY)

```

## 10. Ejercicio 10

```

1 clear
2 clc
3 T=0;
4 D=input('Ingrese el numero de departamentos: ');
5 for K=1:D
6 S=0;
7 M=input('Ingrese el numero de empleados: ');
8 for J=1:M
9 E=input('Ingrese la edad del empleado: ');

```

```

10 S=S+E;
11 end
12 P=S/M;
13 fprintf('El promedio de edad del departamento %d es %.2f \n',K,P)
14 T=T+P;
15 end
16 R=T/D;
17 fprintf('El promedio de edad de la empresa es %.2f \n',R)

```

## 5.4. Solución Ejercicios propuestos Capítulo 3

### 1. Ejercicio 1

```

1 clear
2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 for i=1:N
6 for j=1:M
7 B(i,j)=input('Ingrese los elementos de la matriz: ');
8 end
9 end
10 for i=1:M
11 MAY=0;
12 for j=1:N
13 if B(j,i)>MAY
14 MAY=B(j,i);
15 end
16 end
17 fprintf('El elemento mayor de la columna %d es %d \n',i,MAY)
18 end

```

### 2. Ejercicio 2

```

1 clear
2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 for i=1:N
6 for j=1:M
7 P(i,j)=input('Ingrese los elementos de la matriz: ');
8 end
9 end
10 s=0;
11 for i=1:N
12 for j=1:M
13 if mod(P(i,j),2)==0
14 s=s+P(i,j);
15 end
16 end
17 end
18 fprintf('La sumatoria de los numeros pares de la matriz es %d \n',s)

```

### 3. Ejercicio 3

```

1 clear
2 clc
3 N=input('Ingrese el numero de filas para la matriz: ');
4 M=input('Ingrese el numero de columnas para la matriz: ');
5 i=1;
6 while i<=N
7 j=1;
8 while j<=M
9 a=input('Ingrese los elementos de la matriz: ');
10 if a<10
11 P(i,j)=a;
12 j=j+1;

```



```

13 else
14 fprintf('El elemento no es menor e igual a 10')
15 end
16 end
17 i=i+1;
18 end
19 s=0;
20 for i=1:N
21 for j=1:M
22 if P(i,j)>5
23 P(i,j)=0;
24 else
25 P(i,j)=1;
26 end
27 end
28 end
29 for i=1:N
30 for j=1:M
31 fprintf('El elemento %d,%d es %d \n ',i,j,P(i,j))
32 end
33 end

```

#### 4. Ejercicio 4

```

1 clear
2 clc
3 N=input('Ingrese la dimension de la matriz: ');
4 for i=1:N
5 for j=1:N
6 A(i,j)=input('Ingrese los elementos de la matriz: ');
7 end
8 end
9 for i=1:N
10 for j=1:N
11 if i==j
12 fprintf('El elemento de la diagonal principal es %d \n',A(i,j))
13 end
14 end
15 end

```

#### 5. Ejercicio 5

```

1 clear
2 clc
3 N=input('Ingrese la dimension de la matriz: ');
4 for i=1:N
5 for j=1:N
6 A(i,j)=input('Ingrese los elementos de la matriz: ');
7 B(i,j)=input('Ingrese los elementos de la matriz: ');
8 end
9 end
10 c=0;
11 for i=1:N
12 for j=1:N
13 if i==j
14 if A(i,j)==B(i,j)
15 c=c+1;
16 end
17 end
18 end
19 end
20 if c==N
21 disp('Las diagonales son iguales')
22 else
23 disp('Las diagonales no son iguales')
24 end

```

## 5.5. Solución Ejercicios propuestos Capítulo 4

### 1. Ejercicio 1

Función para leer datos

```
1 function A=leer_datos(N,M)
2 %ingresa datos a una matriz
3 for I=1:N
4 fprintf('Salon %d \n',I)
5 for K=1:M
6 A(I,K)=input('Ingrese la edad de los estudiantes: ');
7 end
8 end
```

Función para obtener el promedio del salón de clase de m estudiantes

```
1 function [p,t]=prom_sal(N,M,A)
2 t=0;
3 for I=1:N
4 p=0;
5 for K=1:M
6 p=p+A(I,K);
7 end
8 t=t+p;
9 p=p/M;
10 fprintf('La edad promedio del salon %d es %.2f \n',I,p)
11
12 end
```

Script para ejecutar las funciones y obtener el promedio del salón y de la escuela

```
1 clear
2 clc
3 T=0;
4 N=input('Ingrese el numero de salones: ');
5 M=input('Ingrese el numero de estudiantes: ');
6 A=leer_datos(N,M);
7 [p,t]=prom_sal(N,M,A);
8 t=t/(N*M);
9 fprintf('El promedio de edad de la escuela es %.2f \n',t)
```

### 2. Ejercicio 2

Función para leer datos

```
1 function V = leer_vector(N)
2 %Lectura de los datos
3 for i=1:N
4 V(i)=input('Ingrese los elementos del vector: ');
5 end
6 end
```

Función para obtener el elemento mayor de un vector

```
1 function MAY=mayor_vector(N,V)
2 MAY=V(1);
3 for I=1:N
4 if V(I)>MAY
5 MAY=V(I);
6 end
7 end
8 fprintf('El elemento mayor es %d \n',MAY)
9 end
```

Script para ejecutar las funciones y obtener el elemento mayor de un vector

```

1 clear
2 clc
3 N=input('Ingrese el numero de elementos del vector: ');
4 V=leer_vector(N);
5 mayor_vector(N,V);

```

### 3. Ejercicio 3

Función para leer datos

```

1 function V = leer_vector(N)
2 %Lectura de los datos
3 for i=1:N
4 V(i)=input('Ingrese los elementos del vector: ');
5 end
6 end

```

Función para clasificar los elementos del vector

```

1 function [c,d,k,p]=clasificar(N,V1,V2)
2 k=1;p=1;
3 for I=1:N
4 if mod(V1(I),2)==0
5 c(k)=V1(I);
6 k=k+1;
7 else
8 d(p)=V1(I);
9 p=p+1;
10 end
11 end
12 for I=1:N
13 if mod(V2(I),2)==0
14 c(k)=V2(I);
15 k=k+1;
16 else
17 d(p)=V2(I);
18 p=p+1;
19 end
20 end

```

Función para imprimir los vectores resultantes

```

1 function v=imprimir_vector(N,x)
2 for i=1:N-1
3 fprintf('El elemento %d del vector es %d \n',i,x(i))
4 end

```

Script para ejecutar las funciones y obtener resultados

```

1 clear
2 clc
3 N=input('Ingrese el numero de elementos del vector: ');
4 V1=leer_vector(N);
5 V2=leer_vector(N);
6 [c,d,k,p]=clasificar(N,V1,V2);
7 imprimir_vector(k,c)
8 imprimir_vector(p,d)

```

### 4. Ejercicio 4

Función para leer la matriz

```

1 function A=leer_matriz(N,M)
2 for I=1:N
3 for K=1:M

```

```

4 A(I,K)=input('Ingrese los elementos de la matriz: ');
5 end
6 end

```

Función para determinar la cantidad de elementos menores e iguales a 5

```

1 function c=menores_cinco(N,M,A)
2 c=0;
3 for I=1:N
4 for K=1:M
5 if A(I,K)≤5
6 c=c+1;
7 end
8 end
9 end
10 fprintf('Existen %d elementos menores iguales a 5 en la matriz \n',c)

```

Función para determinar la suma de filas de la matriz

```

1 function s=suma_filas(N,M,A)
2 for I=1:N
3 S=0;
4 for K=1:M
5 S=S+A(I,K);
6 end
7 fprintf('La suma de la fila %d es %d \n',I,S)
8 end
9
10 end

```

Script para ejecutar las funciones y obtener resultados

```

1 clear
2 clc
3 N=input('i{ngrese el numero de filas: ');
4 M=input('i{ngrese el numero de columnas: ');
5 A=leer_matriz(N,M);
6 menores_cinco(N,M,A);
7 suma_filas(N,M,A);

```

## 5. Ejercicio 5

Función para leer las calificaciones

```

1 function A=leer_calificaciones(N)
2 for I=1:N
3 fprintf('Estudiante %d \n',I)
4 K=1;
5 while K≤3
6 c=input('Ingrese la calificacion del estudiante: ');
7 if c>0 & c≤10
8 A(I,K)=c;
9 K=K+1;
10 else
11 disp('Calificacion incorrecta')
12 end
13 end
14 end

```

Función para determinar el promedio por estudiante

```

1 function [p,t]=prom_estudiantes(N,A)
2 for I=1:N
3 p=0;
4 for K=1:3
5 p=p+A(I,K);

```

```

6 end
7 p=p/3;
8 fprintf('El promedio del alumno %d es %.2f \n', I,p)
9 end

```

Función para determinar el promedio por parcial

```

1 function [p,t]=prom-parcial(N,A)
2 for I=1:3
3 p=0;
4 for K=1:N
5 p=p+A(K, I);
6 end
7 p=p/N;
8 fprintf('El promedio del parcial %d es %.2f \n', I,p)
9 end

```

Script para ejecutar las funciones y obtener resultados

```

1 clear
2 clc
3 T=0;
4 N=input('Ingrese el numero de estudiantes: ');
5 A=leer.calificaciones(N);
6 prom_estudiantes(N,A);
7 prom-parcial(N,A);

```

ISBN: 978-9942-616-27-2



9789942616272