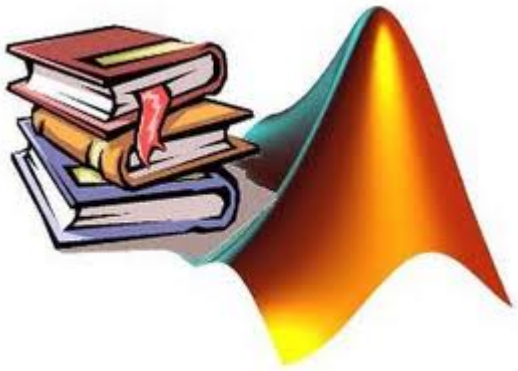
MATLAB®
PROGRAMACIÓN



Escuela Superior Politécnica del Litoral ESPOL
Facultad de Ciencias Naturales y Matemáticas
Departamento de Matemáticas
Guayaquil, Ecuador
2013

Luis Rodríguez Ojeda
lrodrig@espol.edu.ec

MATLAB® PROGRAMACIÓN - Versión 0

Prefacio

Este documento inicial es una contribución bibliográfica para los estudiantes que toman un primer curso de Programación de Computadoras a nivel universitario con el soporte de MATLAB. El contenido de esta obra no tiene pre-requistos, solamente el interés en conocer un lenguaje actual que sea simple y versátil para resolver computacionalmente problemas de diferente nivel de complejidad.

El origen es la experiencia desarrollada por el autor impartiendo estos cursos en el Instituto de Ciencias Matemáticas (Departamento de Matemáticas actualmente) para estudiantes de ingeniería de la ESPOL y contiene parte del material desarrollado mediante ejemplos típicos para decsribir los conceptos algorítmicos en forma práctica y que son la base para resolver problemas más complejos.

El énfasis es el componente algorítmico en la solución de problemas y usa como soporte computacional el programa MATLAB que contiene un amplio repertorio de funciones para aplicaciones numéricas, gráficas y simbólicas y dispone además de un lenguaje de programación muy simple y general. Combinando estos componentes se tiene un instrumento computacional muy potente para resolver problemas en áreas muy diversas en ingeniería, matemáticas y otras ciencias.

En este segundo documento se describe el componente programable de MATLAB el cual requiere usar instrucciones cuya sintaxis es muy simple pero suficiente para resolver problemas más complejos. Este componente es obligatorio para los estudiantes que desean desarrollar su capacidad lógica para enfrentar la solución de problemas para cuya solución no es suficiente el componente interactivo de MATLAB.

Otro objetivo importante al escribir estos documentos es el desarrollo de textos digitales para ser usados en línea, reduciendo el consumo de papel y tinta, contribuyendo así con el cuidado del medio ambiente. Una ventaja adicional de los libros virtuales es la facilidad para actualizar y mejorar continuamente su contenido.

El contenido ha sido compilado en formato pdf. El tamaño del texto en pantalla es controlable, contiene un índice electrónico para facilitar la búsqueda de temas y, dependiendo de la versión del programa de lectura de este formato, se pueden usar las facilidades disponibles para resaltar digitalmente texto, insertar comentarios, notas, enlaces, revisiones, búsqueda por contenido, lectura, etc.

Este documento es de libre uso y distribución. Su realización ha sido factible por el apoyo de la Institución a sus profesores en el desarrollo de las actividades académicas

Luis Rodríguez Ojeda, M.Sc.
lrodrig@espol.edu.ec

Escuela Superior Politécnica del Litoral, ESPOL
Facultad de Ciencias Naturales y Matemáticas
Departamento de Matemáticas
Guayaquil, Ecuador

Contenido

1 Introducción

1.1 Objetivo y requisitos

Proporcionar los conocimientos básicos para usar las facilidades de programación del entorno de MATLAB. Se supondrá que los interesados tienen alguna experiencia previa usando el componente interactivo de MATLAB y al menos el conocimiento elemental de la lógica matemática.

1.2 Metodología

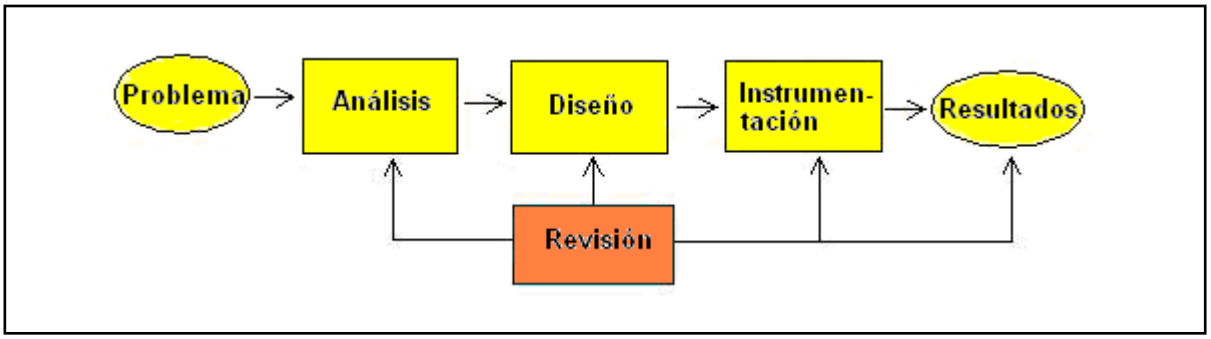
Mediante explicaciones basadas en los ejemplos incluidos en este documento, el usuario puede adquirir en forma progresiva los conocimientos necesarios para programar en MATLAB. El desarrollo de variados ejercicios proporcionará la base para extenderlos y aplicarlos a la resolución de problemas complejos.

Para facilitar el aprendizaje se sugiere abrir dos ventanas en la pantalla de su computador, una con el programa MATLAB y otra con este documento, entonces puede escribir cada ejemplo en la ventana de edición de MATLAB y probarlo interactivamente.

1.3 Un modelo para resolver problemas con el computador

El análisis y diseño de soluciones computacionales es una ciencia que facilita el uso eficiente del poder de las computadoras para resolver problemas.

Para complementar el desarrollo de estas soluciones, es adecuado usar un lenguaje computacional simple, general y eficiente como el que proporciona MATLAB el cual dispone además de gran cantidad de recursos para cálculo numérico, manejo simbólico, visualización y programación en diversas áreas del conocimiento.



Suponer un problema que debemos resolver y que está en nuestro ámbito de conocimiento.

En la etapa de **Análisis** se debe estudiar y entender el problema: sus características, las variables y los procesos que intervienen. Asimismo, se deben definir los datos necesarios y el objetivo esperado. El resultado de esta etapa son las especificaciones detalladas de los requerimientos que en algunos casos se puede expresar mediante modelos matemáticos.

En la etapa de **Diseño** se procede a elaborar los procedimientos necesarios para cumplir con los requerimientos especificados en el análisis, incluyendo fórmulas, tablas, etc. El objeto resultante se denomina algoritmo.

En la etapa de **Instrumentación,** si el problema es simple, se puede obtener la solución interactuando directamente mediante instrumentos disponibles en el entorno computacional. Si el problema es más complejo, deben construirse los programas y organizar los datos necesarios.

Al concluir la etapa de la instrumentación, se realizan las pruebas de los programas y posteriormente la instalación y operación. Debe preverse la necesidad de efectuar mantenimiento y cambios para ajustarlos al entorno en el que se usarán los programas.

Este proceso necesita planificación y alguna sistematización para que se desarrolle en forma eficiente, siendo imprescindible seguir normas y mantener una documentación adecuadas.

Los Instrumentos computacionales modernos tales como MATLAB contienen instrucciones que pueden usarse directamente para resolver muchos problemas. Sin embargo, en general, la resolución de problemas requiere una planificación previa antes de su instrumentación.

2 Algoritmos

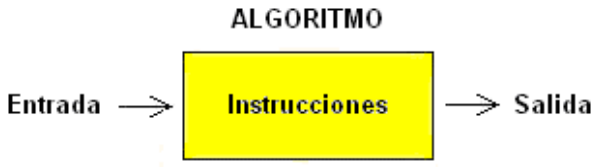
Un algoritmo es una descripción ordenada de las instrucciones que deben realizarse para resolver un problema en un tiempo finito.

Para crear un algoritmo es necesario conocer en forma detallada el problema, las variables, los datos que se necesitan, los procesos involucrados, las restricciones, y los resultados esperados.

La descripción del algoritmo debe orientarse a la instrumentación computacional final. Sin embargo, cuando los problemas son simples, puede omitirse la elaboración física del algoritmo e ir directamente a la codificación en el lenguaje computacional.

2.1 Estructura de un algoritmo

Un algoritmo es un objeto que debe comunicarse con el entorno. Por lo tanto debe incluir facilidades para el ingreso de datos y la salida de resultados.



2.2 Descripción de algoritmos

Para escribir algoritmos se pueden usar diferentes notaciones: textual, gráfica, o simbólica, pero para que una notación sea útil debe poseer algunas características que permitan obtener algoritmos fáciles de entender y aplicar:

1) Las instrucciones deben ser simples para su utilización.

2) Las instrucciones deben ser claras y precisas.

3) Debe incluir suficientes instrucciones para describir la solución de problemas simples y complejos.

4) Preferentemente, las instrucciones deben tener orientación computacional.

Los algoritmos deben ser reproducibles, es decir que al ejecutarse deben entregar los mismos resultados si se utilizan los mismos datos.

2.3 Definiciones

a) Proceso
Conjunto de acciones realizadas al ejecutar las instrucciones descritas en un algoritmo.

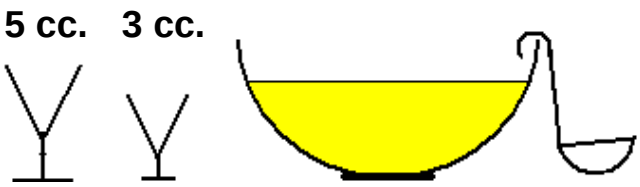
b) Estado
Situación de un proceso en cada etapa de su realización, desde su inicio hasta su finalización. En cada etapa, las variables pueden modificarse.

c) Variables
Símbolos con los que se representan los valores que se producen en el proceso.
Componentes de una variable

Nombre:	Identificación de cada variable
Dominio:	Tipo de datos asociado a una variable
Contenido:	Valor asignado a una variable
Celda:	Dispositivo para el almacenamiento del valor asignado a una variable

Un ejemplo para introducir el concepto de algoritmo

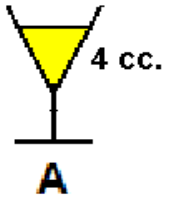
Problema: En el gráfico siguiente suponer que hay un recipiente grande con algún refresco y se requiere tomar una cantidad exacta de 4 cc. Con los instrumentos indicados en el gráfico, describa un algoritmo para obtener esa cantidad. Es posible trasladar el contenido entre recipientes.



Análisis

Sean **A**: recipiente de 5 cc.
B: recipiente de 3 cc.
C: recipiente con el refresco (cantidad suficiente)

Objetivo Propuesto: Colocar los 4 cc. de refresco en el recipiente de 5 cc.



Algoritmo

1. **Llene A**
2. **Vierta A en B hasta llenarlo**
3. **Vierta todo el contenido de B en C**
4. **Vierta todo el contenido de A en B**
5. **Llene A**
6. **Vierta el contenido de A en B hasta llenarlo**
7. **El recipiente A contendrá 4 cc.**

Prueba

Recorrer el algoritmo anotando los valores que toman las variables **A** y **B**

Instrucción	Contenido de A	Contenido de B
1	**5**	**0**
2	**2**	**3**
3	**2**	**0**
4	**0**	**2**
5	**5**	**2**
6	**4**	**3**

Resultado

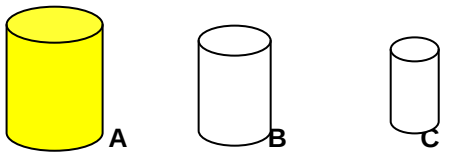
En el recipiente **A** quedarán 4 cc.

Observe los componentes que intervienen en la construcción del algoritmo:

a) Defínción de variables
b) Propuesta del objetivo
c) Lista de instrucciones
d) Prueba del algoritmo
e) Verificación del resultado obtenido

Ejercicio

Se tienen 3 recipientes cilíndricos sin marcas de 12, 7, y 5 galones de capacidad. El recipiente de 12 galones está lleno de combustible. El objetivo es repartir el combustible en partes iguales entre dos personas usando únicamente los tres recipientes. Es posible trasladar el combustible entre recipientes



a) Describa gráficamente el resultado esperado
b) Construya un algoritmo para obtener la solución
c) Cuando haya intentado su solución compare su algoritmo con el que se propone abajo
d) Verifique tabularmente que el algoritmo propuesto produce la solución esperada.

Instrucción	A	B	C

Una solución

1 Vierta A en C
2 Vierta C en B
3 Vierta A en B hasta llenar
4 Vierta B en C hasta llenar
5 Vierta C en A
6 Vierta B en C
7 Vierta A en B hasta llenar
8 Vierta B en C hasta llenar
9 Vierta C en A
10 Vierta B en C
11 Vierta A en B hasta llenar
12 Vierta B en C hasta llenar
13 Vierta C en A

3 Desarrollo de programas en MATLAB

Un programa es la descripción de un algoritmo en un lenguaje computacional. Para escribir un programa es necesario conocer las reglas de sintaxis y semántica del lenguaje de programación y haber estructurado el algoritmo para resolver el problema de interés, incluyendo las variables y los procesos requeridos.

3.1 Algunos elementos básicos para escribir programas

Operadores aritméticos: **+ - * / \ ^**

Símbolos especiales: **[ ], ( ), %**

Funciones matemáticas: **sin, cos, tan,…exp,log,log10,sqrt,…abs,fix,mod,sign,…**

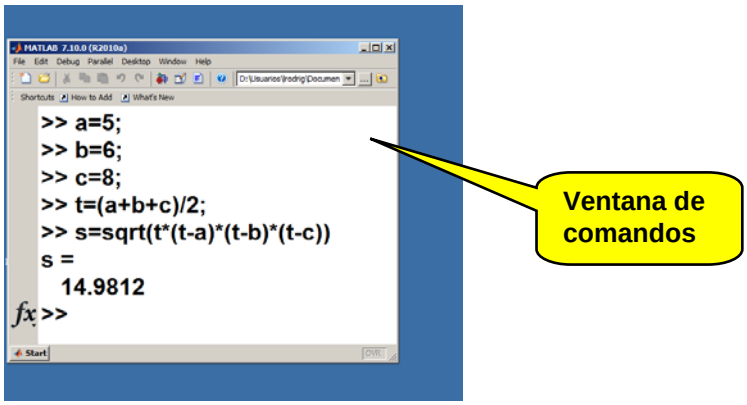
Símbolos numéricos especiales: **pi, eps, Inf, NaN, …**

3.2 Transición del uso interactivo de MATAB en la ventana de comandos hacia la edición y prueba de programas.

Ejemplo: Calcule el área de un triángulo cuyos lados miden 5, 6, 8 cm.

3.2.1 Solución en modo interactivo en la ventana de comandos

Digite, ingrese y observe el resultado de la ejecución de cada instrucción



3.2.2 Solución mediante un archivo que contiene los comandos

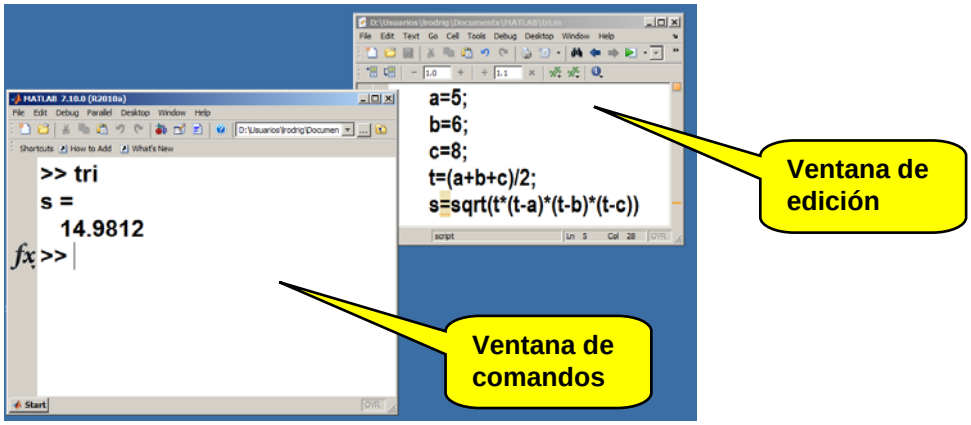
En MATLAB un archivo de comandos o script constituye un programa

Para crear un programa se debe abrir la ventana de edición con el botón o ícono **new scrip** ubicado arriba a la izquierda en la ventana de MATLAB.

Esta ventana de edición es una página en la que puede escribir texto libremente con la facilidades típicas de un procesador de texto. Escriba los comandos y guarde el documento en un archivo con algún nombre. Para ejecutar el programa se debe escribir en la ventana de comandos el nombre del programa

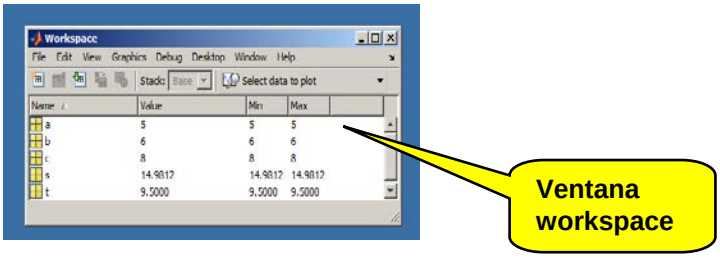
Realice la siguiente práctica:

1) Abra una ventana de edición
2) Digite las instrucciones
3) Almacene en un archivo con algún nombre. Ejemplo: **tri**
4) Ejecute las instrucciones desde el archivo, escribiendo el nombre del programa. Ejemplo: **tri**



Para realizar cambios en el programa que está abierto en la ventana de edición, posicione el cursor en el lugar respectivo, realice los cambios, almacene el archivo y ejecute nuevamente.

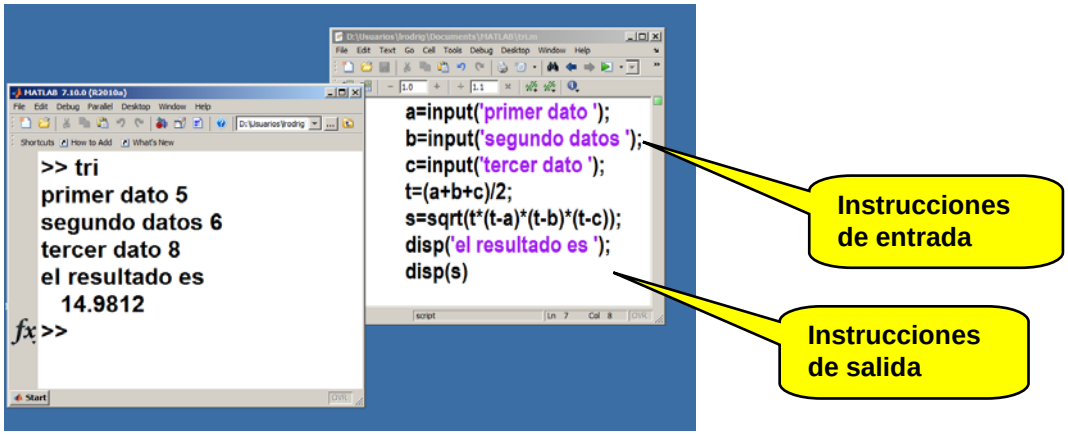
Para visualizar el mapa de memoria abra la ventana de trabajo **workspace**



3.2.3 Solución agregando instrucciones de entrada y salida al programa

Se puede postergar el ingreso de los datos para el momento de su ejecución, evitando que estén asignados dentro del programa. Para esto se usa la instrucción **input**. Igualmente, la salida conviene distringuirla con una instrucción especial, **disp**

Haga los cambios siguiendo el ejemplo, almacene y ejecute el nuevo programa. Realice la interacción de ingreso de los datos. Realice varias pruebas con el mismo programa



Para realizar pruebas con un programa, es muy conveniente que los programas sean independientes de los datos. Esto se consigue con el uso de las instrucciones de entrada de datos.

Adicionalmente, es una buena norma de programación el insertar algún comentario para identificar el problema que se está resolviendo. Puede incluir comentarios en su programa iniciándolos con el símbolo **%**

4 Instrucciones básicas para algoritmos y programas

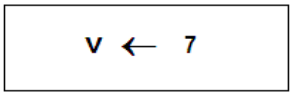
Cada instrucción de MATLAB se describirá mediante ejemplos y relacionándola con su descripción en lenguaje natural y su representación gráfica en los diagramas de flujo

4.1 Instrucción de asignación

Lenguaje natural

Almacene en **v** el valor **7**

Diagrama de flujo



MATLAB

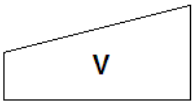
```
v = 7
```

4.2 Instrucción para ingreso de datos

Lenguaje natural

Ingrese un dato para **V**

Diagrama de flujo



MATLAB

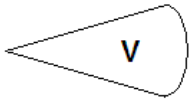
```
v = input('Ingrese un dato ');
```

4.3 Instrucción para salida de resultados

Lenguaje natural

Muestre el contenido de **V**

Diagrama de flujo



MATLAB

```
disp( v );
```

Ejemplo. Calcule el pago semanal que recibe un obrero dados los siguientes datos: horas trabajadas, tarifa por hora, descuentos.

Análisis

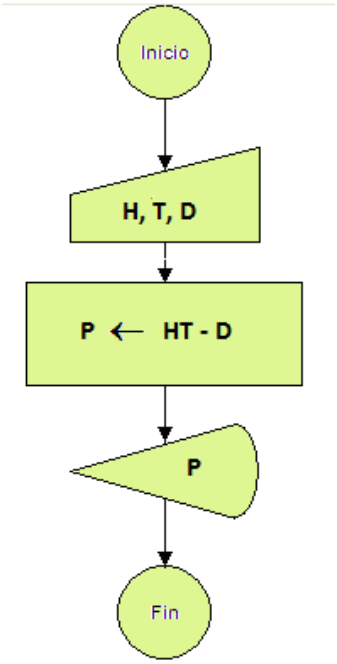
Datos: **H:** Cantidad de horas trabajadas en la semana
T: Tarifa de pago por hora.
D: Descuentos en la semana.
Resultado: **P:** Valor a pagar (valor ganado menos descuentos)

Algoritmo

Lenguaje natural

Ingrese un valor para **H**
Ingrese un valor para **T**
Ingrese un valor para **D**
Calcule **P = H*T - D**
Muestre el valor de **P**

Diagrama de Flujo



MATLAB

```
%Pago semanal
h=input('Horas trabajadas ');
t=input('Tarifa por hora ');
d=input('Descuentos ');
p=h*t-d;
disp('Valor a pagar');
disp(p);
```

Ejemplo. Calcule el área y el volumen de un cilindro dados su radio y altura.

Análisis

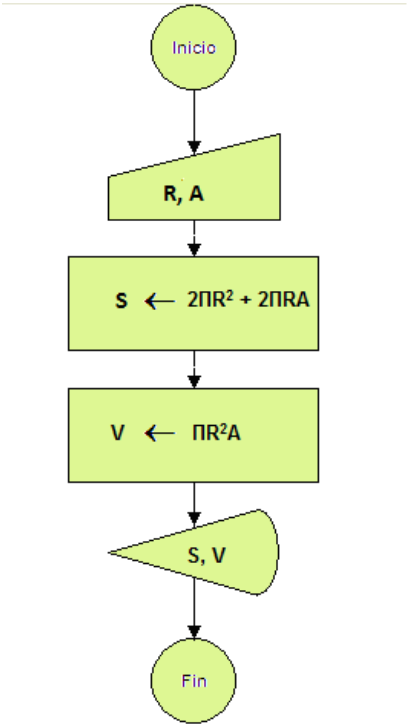
Datos:	**R:** Radio	
	A: Altura	
Resultados:	**S:** Area	
	V: Volumen	
Fórmulas:	**S** = $2\Pi R^2 + 2\pi RA$	(área)
	V = $\Pi R^2 A$	(volumen)

Algoritmo

Lenguaje natural

Ingrese un valor para **R**
Ingrese un valor para **A**
Calcule **S** = $2\Pi R^2 + 2\Pi RA$
Calcule **V** = $\Pi R^2 A$
Muestre el valor de **S**
Muestre el valor de **V**

Diagrama de Flujo



MATLAB

```
%Area y volumen de un cilindro
r=input('Radio ');
a=input('Altura ');
s = 2*pi*r^2 + 2*pi*r*a;
v = pi*r^2*a;
disp('Area');
disp(s)
disp('Volumen');
disp(v);
```

Ejemplo. Calcular el valor de una compra si los datos son la cantidad comprada y el precio unitario. Incluir el impuesta del IVA.

Análisis

Variables

Datos:	**C:** contendrá la cantidad de artículos comprados
	P: contendrá el precio unitario de los artículos
Resultado	**T:** contendrá el valor total de la compra

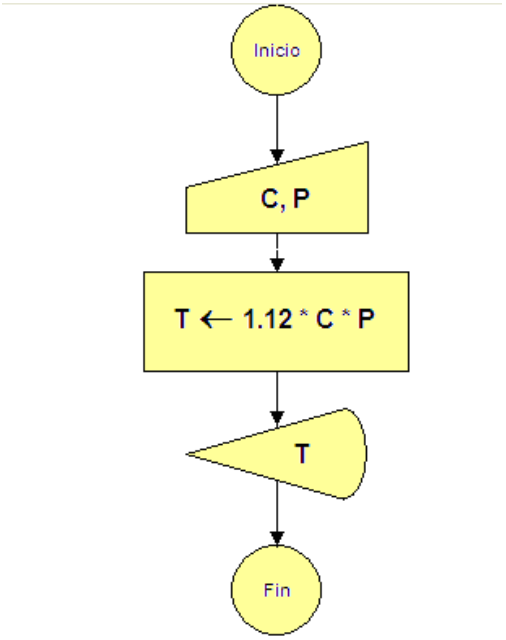
Impuesto del IVA: 0.12 del valor de la compra

Algoritmo

Lenguaje natural

Ingrese un valor para **C**
Ingrese un valor para **P**
Almacene en **T** el resultado de **1.12*C*P**
Muestre el valor de **T**

Diagrama de Flujo



Ingreso de datos

Agregar 12% del IVA

Salida de resultados

MATLAB

```
% Cálculo del valor de una compra
c = input('Ingrese la cantidad de artículos ');
p = input('Ingrese el precio unitario ');
t = 1.12 * c * p;
disp('Valor de la compra');
disp(t);
```

4.3.1 Ejercicios de programación con las instrucciones básicas

Para cada ejercicio escriba y pruebe un programa con MATLAB

1.- Dados el radio y altura de un cilindro calcule el área total y el volumen

2.- Se tiene un recipiente cilíndrico con capacidad de x litros. Su altura es h metros. Determine el diámetro de la base

3.- Calcule el promedio de las tres calificaciones de las lecciones que obtuvo un estudiante

4.- Dadas las tres dimensiones de un recipiente rectangular, calcule su volumen

5. La siguiente fórmula proporciona el enésimo término **u** de una progresión aritmética:
u = a + (n − 1) r en donde **a** es el primer término, **n** es el la cantidad de términos y **r** es la razón entre dos términos consecutivos
Calcular el valor de **r** dados **u, a, n**

6. En el ejercicio anterior, calcular el valor de: **n** dados **u, a, r**

7.- Un modelo de crecimiento poblacional está dado por: $n = 5t + 2e^{0.1t}$, en donde n es el número de habitantes, t es tiempo en años.

Calcule el número de habitantes que habrían en tres años consecutivos

8- Un ingeniero desea tener una cantidad de dólares acumulada en su cuenta de ahorros para su retiro luego de una cantidad de años de trabajo. Para este objetivo planea depositar un valor mensualmente. Suponga que el banco acumula el capital mensualmente mediante la siguiente fórmula:

$$\mathbf{A = P\left[\frac{(1+x)^n - 1}{x}\right]}, \text{ en donde}$$

A: Valor acumulado
P: Valor de cada depósito mensual
n: Cantidad de depósitos mensuales
x: Tasa de interés mensual

Calcule el valor acumulado ingresando como datos valores para P, n, x

9.- Una empresa produce fertilizantes. Cada mes el ingreso por ventas en miles de dólares se describe con v = 0.4x (30 - x) mientras que el costo de producción en miles de dólares es v = 5 + 10 ln(x), siendo x la cantidad producida en toneladas, 1<x<30.

Determine el valor del ingreso neto, dado un valor para x.

4.4 Funciones para aritmética con enteros

Las funciones FIX y MOD

FIX: Trunca los decimales de un número y entrega la parte entera.

```
>> c=fix(20/6)
c =
    3
```

MOD: Entrega el residuo de la división entera entre dos números

```
>> r=mod(20,6)
r =
    2
```

Ejemplo. Dado un número entero de dos cifras, sumar sus cifras.

```
%Sumar las cifras de un número de dos dígitos
n=input('Ingrese un numero de dos cifras ');
d=fix(n/10);
u=mod(n,10);
s=d+u;
disp('Suma ');
disp(s);
```

Ejemplo. Dado un número entero de dos cifras, mostrarlo con las cifras en orden opuesto.

```
n=input('Ingrese un numero de dos cifras ');
d=fix(n/10);
u=mod(n,10);
r=10*u+d;
disp('Numero invertido ');
disp(r);
```

Ejemplo. Dado un número entero (cantidad de dólares), mostrar el equivalente usando la menor cantidad de billetes de 100, 50, 20, 10, 5 y 1.

```
x=input('Ingrese la cantidad de dinero ');
c=fix(x/100);
disp(c);
disp('billetes de 100');
x=mod(x,100);
c=fix(x/50);
disp(c);
disp('billetes de 50');
x=mod(x,50);
c=fix(x/20);
disp(c);
.
.                                    (continuar)
.
```

4.4.1 Ejercicios de programación con las funciones fix y mod

Para cada ejercicio escriba y pruebe un programa con MATLAB

1.- Escriba un programa para ingresar un número entero (dias). Determine y muestre el equivalente en meses y días sobrantes. Por simplicidad suponga que cada mes tiene 30 días. Use **fix** para convertir a meses y **mod** para obtener días sobrantes**.** Ej. 198 días equivalen a 6 meses y 18 días

2.- Lea dos números de tres cifras cada uno. Sume la cifra central del primer número con la cifra central del segundo número y muestre el resultado.

3.- Dado un número entero de tres cifras. Determine si la cifra central es igual a la magnitud de la diferencia entre la primera y la tercera cifras

4.5 Decisiones

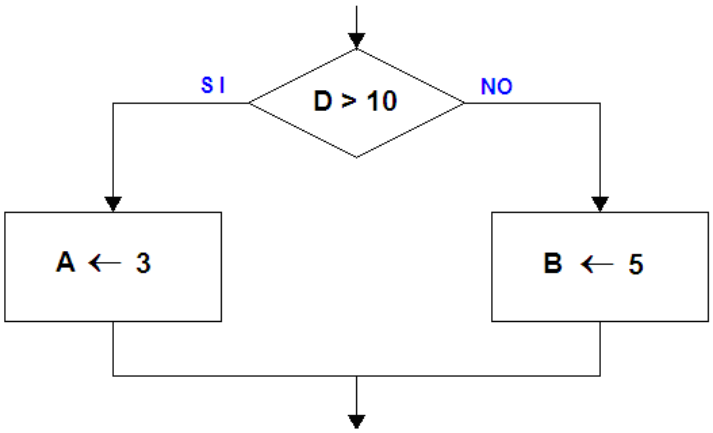
Las decisiones son instrucciones que se usan para describir la ejecución de instrucciones en forma opcional dependiendo del resultado de una expresión cuyos posibles resultados son: verdadero o falso

4.5.1 Decisiones con dos opciones

Lenguaje natural

Si **D > 10** almacene en **A** el valor **3** sino almacene en **B** el valor **5**

Diagrama de flujo



MATLAB

```
if   d > 10
     a = 3;
else
     b = 5;
end
```

4.5.2 Operadores relacionales y lógicos

Para escribir una condición con cuyo resultado se realizará alguna de las opciones, se pueden usar operadores relacionales y conectores lógicos.

Matemáticas	MATLAB
Operadores relacionales	
<, >, ≤, ≥, =, ≠	<, >, <=, >=, ==, ~=
Conectores lógicos	
∧, ∨, ¬	&, |, ~

Ejemplo. Calcular el total que una persona debe pagar en un almacén de llantas. El precio de cada llanta es de $80 si se compran menos de 5 llantas y de $70 si se compran 5 o más.

Análisis

Datos: **N:** Cantidad de llantas

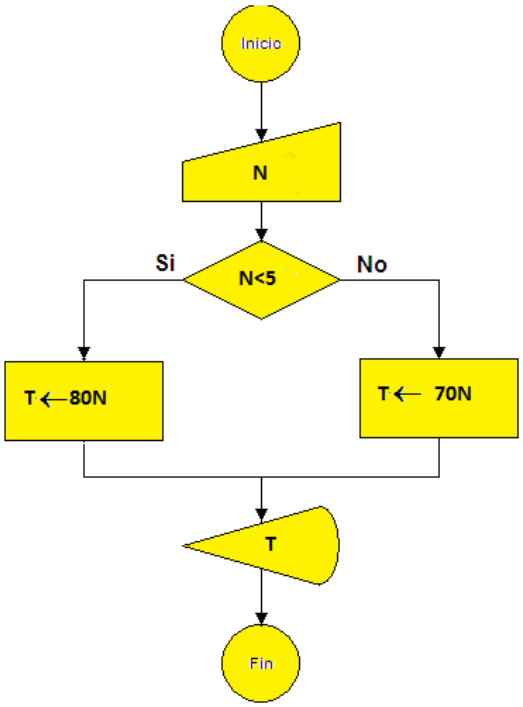
Resultado: **T:** Valor a pagar (Depende de la cantidad de llantas)

Algoritmo

Lenguaje natural

Ingrese un valor para **N**
Si **N < 5** Calcule **T = 80*N**
Caso contrario Calcule **T = 70*N**
Muestre el valor de **T**

Diagrama de Flujo



Ingreso de datos

Decisión para elegir entre las dos opciones para calcular el valor a pagar

Salida del resultado

MATLAB

```
%Compra de llantas con descuento
n=input('Cantidad de llantas ');
if n < 5
   t = 80*n;
else
   t = 70*n;
end
disp('Valor a pagar');
disp(t)
```

Ejemplo. Calcule el pago semanal que recibe un obrero dados los Siguientes datos: horas trabajadas, tarifa por hora, descuentos. Si la cantidad de horas es mayor a 40, la tarifa es mayor en 50%.

Análisis

Datos: **H:** Cantidad de horas trabajadas en la semana
T: Tarifa de pago por hora.
D: Descuentos en la semana.
Resultado: **P:** Valor a pagar (valor ganado menos descuentos)

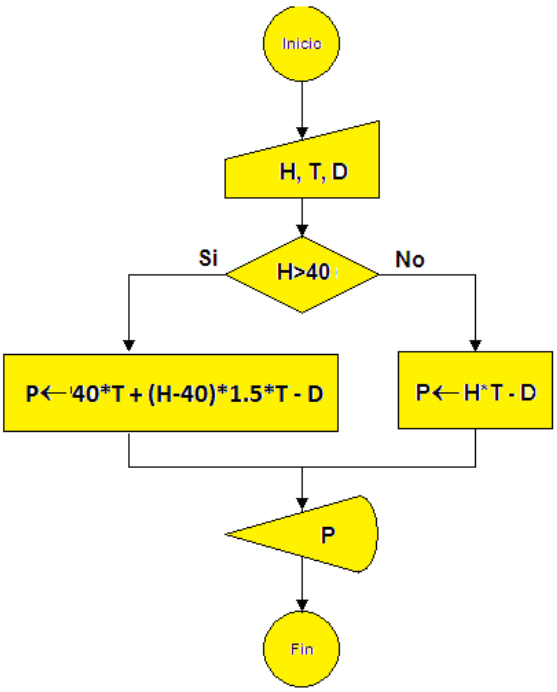
Las horas trabajadas son el exceso de las 40 horas semanales

Algoritmo

Lenguaje natural

Ingrese un valor para **H**
Ingrese un valor para **T**
Ingrese un valor para **D**
Si **H > 40** Calcule **P = 40*T + (H-40)*1.5*T - D**
Caso contrario Calcule **P = H*T - D**
Muestre el valor de **P**

Diagrama de Flujo



MATLAB

```
%Pago semanal
h=input('Horas trabajadas ');
t=input('Tarifa por hora ');
d=input('Descuentos ');
if h>40
   p=40*t+(h-40)*1.5*t-d;
else
   p=h*t-d;
end
disp('Valor a pagar');
disp(p)
```

Ejemplo. Dado un número entero determinar si es un número par o impar.

```
%Determinar si un número es para o impar
x=input('Ingrese un numero ');
r=mod(x, 2);
if r == 0
   disp('Par');
else
   disp('Impar');
end
```

Ejemplo. Dados dos números enteros, determinar si uno es múltiplo del otro

```
%Determinar si un número es múltiplo de otro
a=input('Ingrese el primer numero ');
b=input('Ingrese el segundo numero ');
if mod(a,b) == 0 | mod(b,a) == 0
   disp('son multiplos');
else
   disp('no son multiplos');
end
```

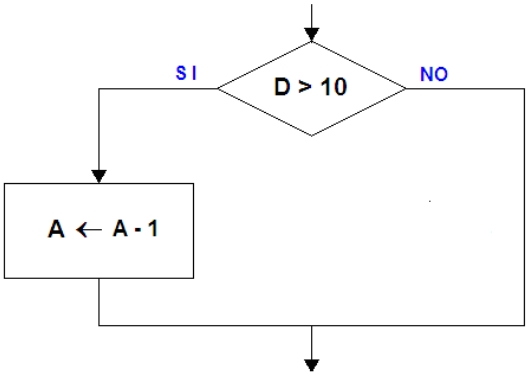
4.5.3 Decisiones con una opción

Normalmente se usa la decisión simple o con una opción, para modificar el valor de una variable que ha sido previamente asignado

Lenguaje natural

Si **D > 10** cambie el valor de **A** restándole **1**

Diagrama de flujo



MATLAB

```
if   d > 10
     a = a - 1
end
```

El ejemplo de las llantas se lo puede resolver con un algoritmo con una decisión con una opción

Ejemplo. Calcular el total que una persona debe pagar en un almacén de llantas. El precio de cada llanta es de $80 si se compran menos de 5 llantas y de $70 si se compran 5 o más.

Análisis

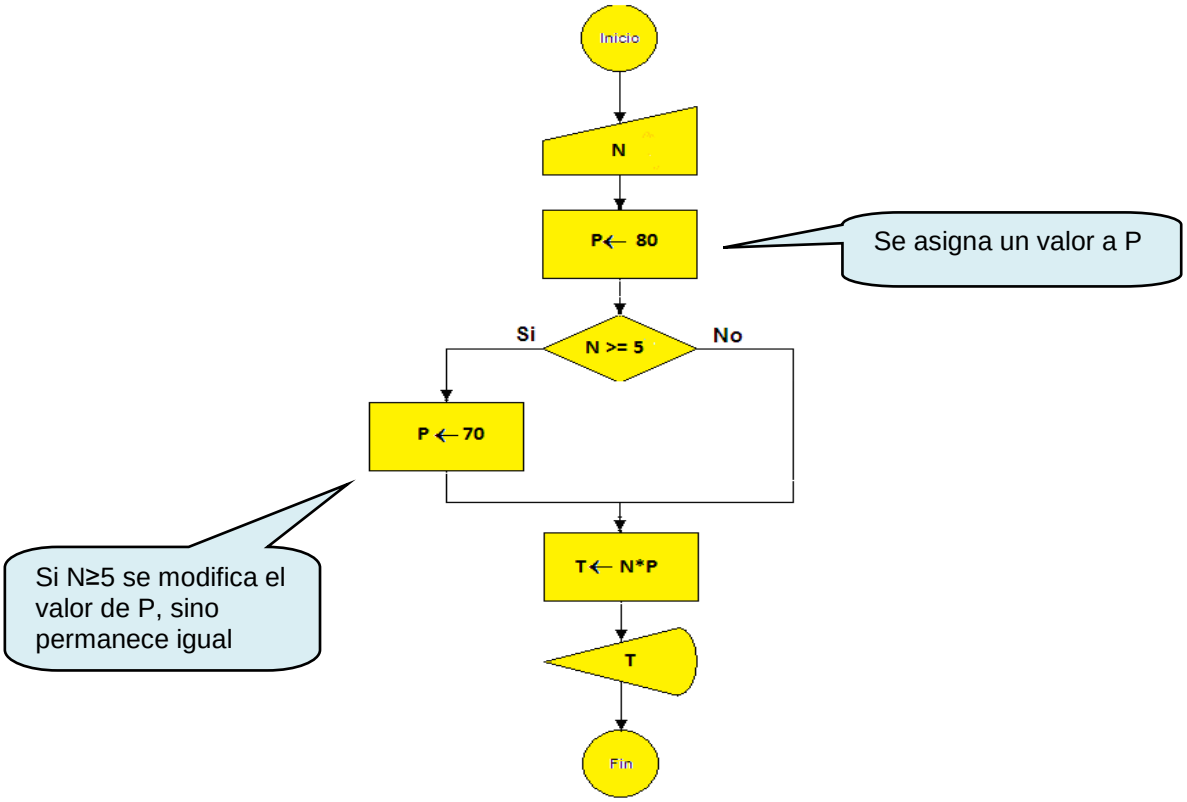
Dato: **N:** Cantidad de llantas
P: Precio unitario (constante que depende de la cantidad de llantas)
Resultado: **T:** Valor a pagar

Algoritmo

Lenguaje natural

Ingrese un valor para **N**
Asigne a **P** el valor **80**
Si **N >= 5** Asigne a **P** el valor **70**
Calcule **T = N*P**
Muestre el valor de **T**

Diagrama de Flujo



MATLAB

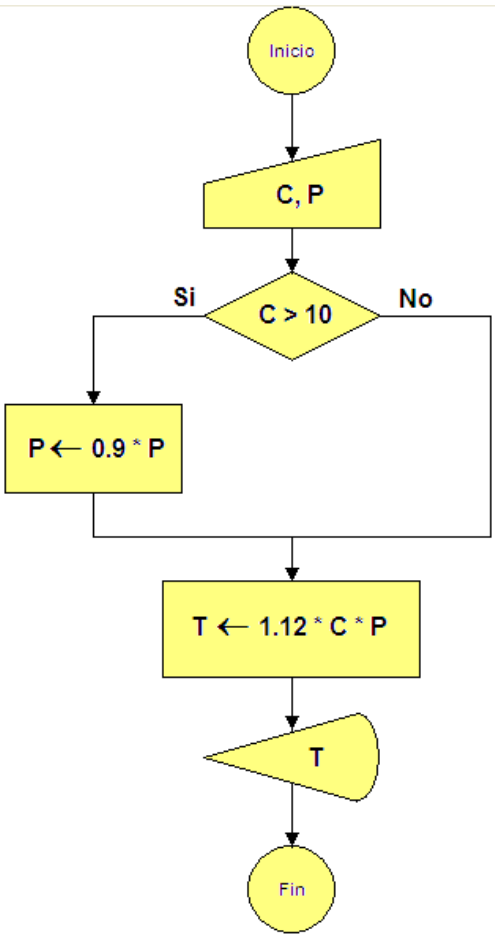
```
%Compra de llantas con descuento
n=input('Cantidad de llantas ');
p=80;
if n >= 5
   p = 70;
end
t=n*p;
disp('Valor a pagar');
disp(t)
```

Ejemplo. Calcular el valor de una compra, dada la cantidad de artículos y el precio unitario. Si la cantidad comprada es mayor a 10 el precio unitario se reduce en 10%. En el resultado se debe incluir el 12% de impuesto del IVA.

Análisis

C: contendrá la cantidad de artículos comprados
P: contendrá el precio unitario de los artículos
T: contendrá el valor total de la compra

Algoritmo



Ingreso de datos

Decisión para aplicar el descuento

Reducir 10% al precio unitario

Agregar 12% del IVA

Salida de resultados

MATLAB

```
% Compra con descuento
c = input('Ingrese la cantidad de artículos ');
p = input('Ingrese el precio unitario ');
if c > 10
   p = 0.9 * p;
end
t = 1.12 * c * p;
disp('Valor de la compra');
disp(t);
```

4.5.4 Decisiones múltiples

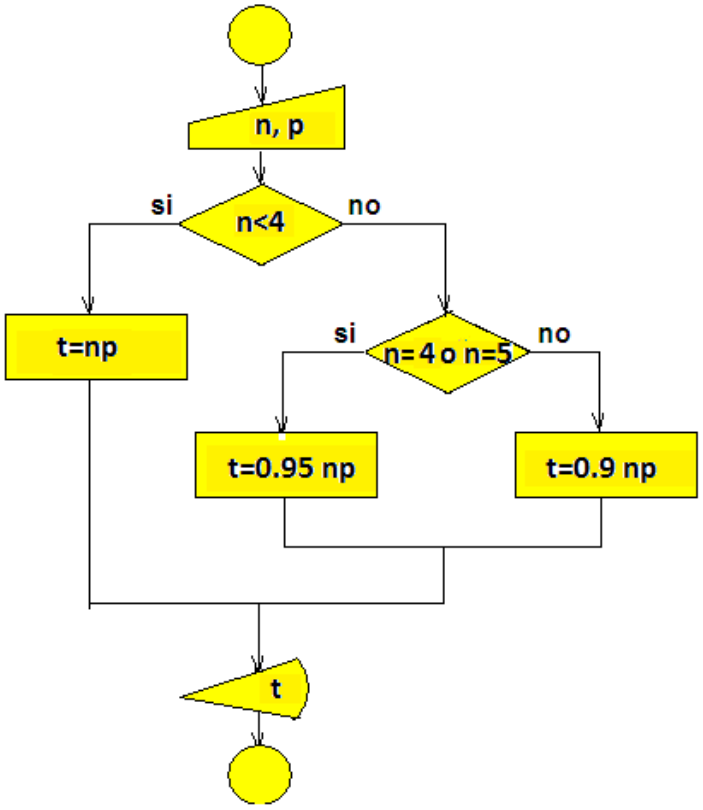
Para describir la selección de una opción entre varias opciones posibles

Ejemplo. Suponga que un local vende llantas según la siguiente política:

Cantidad	Precio
Menos de 4	Sin descuento
4 o 5	5% de descuento
Más de 5	10% de descuento

Lea el número de llantas de una compra y el precio unitario. Muestre el valor que debe pagar

Algoritmo



MATLAB.

```
% Compra de llantas con opciones de descuento
n=input('Cantidad de llantas ');
p=input('Precio unitario ');
if n<4
        t=n*p;
else
        if n==4 | n==5
                t=0.95*n*p;
        else
                t=0.9*n*p;
        end
end
disp('Valor a pagar ');
disp(t);
```

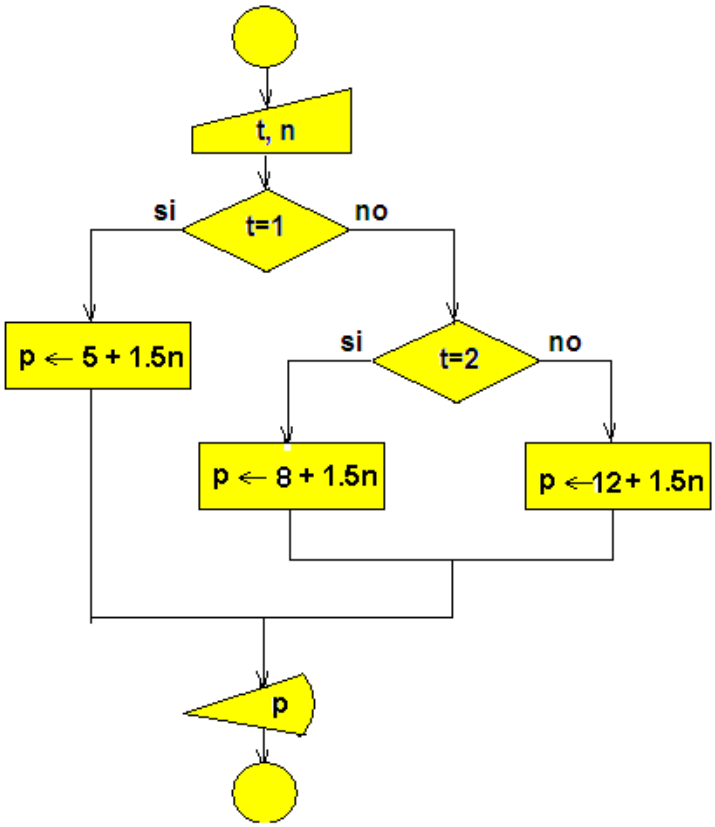
En todos estos ejemplos se supone que los datos ingresados son correctos.

Ejemplo. El valor de una pizza depende de su tamaño:

Tamaño	Precio
1	$5
2	$8
3	$12

Cada ingrediente adicional cuesta $1.5. Lea el tamaño de la pizza y el número de ingredientes y muestre el valor que debe pagar

Algoritmo



MATLAB.

```
%Compra de una pizza
t=input('Tamaño de la pizza ');
n=input('Número de ingredientes ');
if t==1
        p=5+1.5*n;
else
        if t==2
                p=8+1.5*n;
        else
                p=12+1.5*n;
        end
end
disp('Valor a pagar ');
disp(p);
```

¿Qué ocurriría si el dato del tamaño de la pizza no fuese correcto?

En la siguiente solución se considera la posibilidad de que el dato ingresado sea incorrecto. En este caso el valor a pagar será cero

```
%Compra de una pizza
t=input('Tamaño de la pizza ');
n=input('Número de ingredientes ');
if t==1
   p=5+1.5*n;
else
   if t==2
      p=8+1.5*n;
   else
      if t==3
         p=12+1.5*n;
      else
         p=0;
      end
   end
end
disp('Valor a pagar ');
disp(p);
```

4.5.5 La instrucción SWITCH

Cuando existen decisiones múltiples, una estructura alternativa es la instrucción **SWITCH** de **MATLAB**

```
switch variable
        case valor,
                instrucciones
        case valor,
                instrucciones
        case valor,
                instrucciones
         ...

        otherwise,
        instrucciones
end
```

Para usar esta instrucción la **variable** debe ser de tipo cardinal, es decir que se pueda contar, por lo tanto no puede ser de tipo real

La sección **otherwise** es opcional y es ejecutada cuando no se realiza ninguno de los casos incluidos en la instrucción **switch**.

Se pueden omitir las comas luego del valor de cada caso si se escriben las instrucciones en la siguiente línea después de **case**

Ejemplo. Resuelva el problema de la pizza usando la instrucción SWITCH

```
%Compra de una pizza
t=input('tamaño de la pizza ');
n=input('número de ingredientes ');
switch t
   case 1
         p=5+1.5*n;
   case 2
         p=8+1.5*n;
   case 3
         p=12+1.5*n;
   otherwise
       p=0;
end
disp('Valor a pagar ');
disp(p);
```

Esta solución incluye la posibilidad de que el dato ingresado sea incorrecto. En este caso el valor a pagar será cero.

Ejemplo. Dado un número entero entre 1 y 5 mostrar el día de la semana.

Uso de la instrucción **switch.**

d: Contendrá el número del día
La salida es un mensaje

Examine las diferencias en las siguientes soluciones

```
%Día de la semana
d=input('Numero del dia ');
switch d
   case 1, disp('Lunes');
   case 2, disp('Martes');
   case 3, disp('Miercoles');
   case 4, disp('Jueves');
   case 5, disp('Viernes');
end
```

```
d=input('Numero del dia ');
switch d
   case 1, disp('Lunes');
   case 2, disp('Martes');
   case 3, disp('Miercoles');
   case 4, disp('Jueves');
   case 5, disp('Viernes');
   otherwise, disp('Error');
end
```

```
d=input('Numero del dia ');
switch d
   case 1, disp('Lunes');
   case 2, disp('Martes');
   case 3, disp('Miercoles');
   case 4, disp('Jueves');
   case 5, disp('Viernes');
   case {6,7}, disp('Feriado');
   otherwise, disp('Error');
end
```

4.5.6 Ejercicios de programación con decisiones

Para cada ejercicio escriba y pruebe un programa en la ventana de edición de MATLAB

1.- Dados el radio y altura de un cilindro, si la altura es mayor al radio calcule y muestre el valor del volumen, caso contrario muestre el mensaje: 'Error'

2.- Dadas las dimensiones de un bloque rectangular, calcule y muestre el área de la cara de mayor dimensión.

3.- Lea un número de dos cifras. Determinar si la suma de ambas cifras es un número par o impar. Muestre un mensaje

4.- Lea un número. Determine si es entero y múltiplo de 7

5.- Lea la cantidad de Kw que ha consumido una familia y el precio por Kw. Si la cantidad es mayor a 700, incremente el precio en 5% para el exceso de Kw sobre 700. Muestre el valor total a pagar.

6.- Lea un valor de temperatura **t** y un código **p** que puede ser **1** o **2**. Si el código es **1** convierta la temperatura **t** de grados **f** a grados **c** con la fórmula **c=5/9(t-32).** Si el código es **2** convierta la temperatura **t** de grados **c** a **f** con la fórmula: **f=32+9t/5**

7.- Dadas las tres calificaciones de un estudiante, encuentre y muestre su calificación final.

8.- Dados los tres lados de un triángulo determine su tipo: Equilátero, Isòsceles, o Escaleno

9.- El número de pulsaciones que debe tener una persona por cada 10 segundos de ejercicio aeróbico se calcula con la fórmula:

Género femenino (1): número de pulsaciones = (220 - edad en años)/10
Género masculino (2): número de pulsaciones = (210 - edad en años)/10

Lea la edad y el género y muestre el número de pulsaciones

10.- El valor de una pizza depende de su tamaño:

Tamaño	precio
1	$5
2	$8
3	$12

Cada ingrediente adicional cuesta $1.5. Lea el tamaño de la pizza y el número de ingredientes y muestre el valor que debe pagar

11.- Dado el peso P en Kg. de una persona y su talla T en mt. El índice de masa corporal IMC se calcula con la fórmula P/T^2
Determine el nivel de sobrepeso de una persona de25 a 34 años.

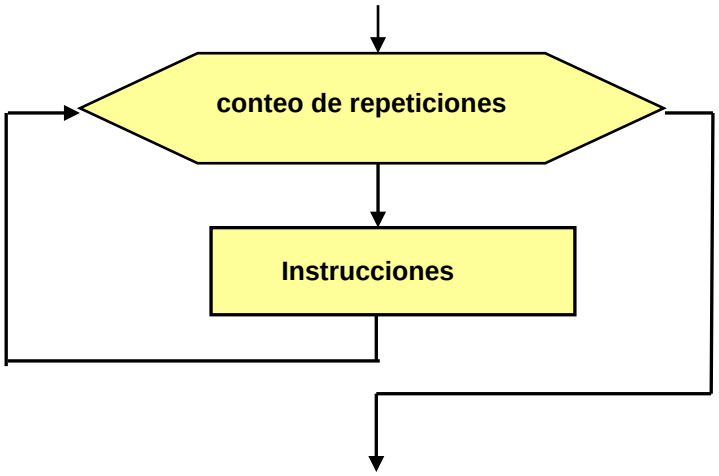
IMC	Nivel
Menos de 18.5	Bajo peso
18.5 a 25.5	Peso Normal
25.5 a 30.5	Sobrepeso grado 1
30.5 a 35.5	Sobrepeso grado 2
35.5 a 40.5	Obesidad grado 3
Más de 40.5	Obesidad grado 4

4.6 Repeticiones

Permiten describir algoritmos en los cuales algunas instrucciones debe ejecutarse varias veces para obtener el resultado esperado.

4.6.1 El ciclo FOR

Representación gráfica del ciclo for



Las instrucciones se realizarán la cantidad de veces que indique el conteo de repeticiones.

La salida del ciclo se producirá al completarse los valores de la variable.

El ciclo for en MATLAB

```
for   conteo de repeticiones

        instrucciones que se repiten

end
```

El conteo de repeticiones se lo puede definir mediante una variable para la que se especifica el valor inicial, el valor final y el incremento, con la siguiente sintaxis:

variable = inicio : incremento: final

Si no se escribe el valor del incremento, se supondrá que es **1**.

variable = inicio : final

Ejemplo. Escriba un programa que muestre 4 veces un nombre

```
for  i = 1: 4
   disp('Simón Bolívar');
end
```

Salida

```
Simón Bolívar
Simón Bolívar
Simón Bolívar
Simón Bolívar
```

Ejemplo. Liste los números naturales entre **1** y **n**, siendo **n** un dato

```
n=input('Ingrese el valor final ');
for  i = 1: n
   disp(i);
end
```

Salida

```
Ingrese el valor final  5
        1
        2
        3
        4
        5
```

Se puede especificar otro valor para el incremento en el conteo:

Ejemplo. Listar los números naturales impares entre 1 y n

```
n=input('Ingrese el valor final ');
for  i= 1: 2: n
   disp(i);
end
```

Salida

```
Ingrese el valor final  5
        1
        3
        5
```

La secuencia puede incluir valores decimales. Ejemplo: **0, 0.1, 0.2, 0.3, …, 2.0**

```
for  i = 0: 0.1: 2
```

Puede ser decreciente. Ejemplo: **5, 4, 3, 2, 1**

```
for  i = 5: -1: 1
```

Puede ser una lista arbitraria. Ejemplo: **7, 4, 5, 6, 8, 2**

```
for  i = [7, 4, 5, 6, 8, 2]
```

Ejemplo. Dado un número entero, liste las raíces cuadradas de los enteros impares entre 1 y el dato dado.

```
n=input('Ingrese el valor final ');
for  x = 1: 2: n
   r=sqrt(x);
   disp([x,r]);
end
```

Salida

```
Ingrese el valor final  9
    1    1
    3    1.7321
    5    2.2361
    7    2.6458
    9    3
```

Ejemplo. Suma de los cuadrados de los primeros **n** números naturales

```
n=input('Ingrese el valor final ');
s=0;
for  i = 1: n
   c=i^2;
   s=s+c:
end
disp('La suma es');
disp(s);
```

Salida

```
Ingrese el valor final  5

La suma es
    55
```

Ejemplo. Muestre el promedio de n datos ingresados desde el teclado

```
n=input('Cantidad de datos ');
s=0;
for  i = 1: n
   x=input('Ingrese un dato ');
   s=s+x:
end
p=s/n;
disp('El promedio es');
disp(p);
```

```
Cantidad de datos  5
 Ingrese un dato  20
 Ingrese un dato  24
 Ingrese un dato  32
 Ingrese un dato  41
 Ingrese un dato  26

 El promedio es
   28.6
```

Ejemplo. Elija y muestre la mayor calificación de **n** datos ingresados desde el teclado

```
n=input('Cantidad de datos ');
m=0;
for  i = 1: n
   x=input('Ingrese un dato ');
   if x>m
     m=x:
   end
end
disp('El mayor valor es');
disp(m);
```

```
Cantidad de datos  5
Ingrese un dato  70
Ingrese un dato  60
Ingrese un dato  80
Ingrese un dato  65
Ingrese un dato  72

El mayor valor es
  80
```

Ejemplo. Elija y muestre la menor calificación de **n** datos ingresados desde el teclado

```
n=input('Cantidad de datos ');
m=Inf;
for  i = 1: n
   x=input('Ingrese un dato ');
   if x<m
     m=x:
   end
end
disp('El menor valor es');
disp(m);
```

Salida

```
Cantidad de datos  5
Ingrese un dato  70
Ingrese un dato  60
Ingrese un dato  80
Ingrese un dato  65
Ingrese un dato  72

El menor valor es
   60
```

4.6.2 Números aleatorios en MATLAB

La función **rand** genera un número aleatorio real en el rango [0, 1)

```
>> rand
```

Se obtendrá un resultado entre 0 y 0.9999….

Se puede modificar el rango

Ejemplo. Obtener un entero entre 0 y 9

```
>> fix(10*rand)
```

Ejemplo. Obtener un entero entre 1 y 10

```
>> fix(10*rand)+1
```

Ejemplo. Obtener un número aleatorio entero entre 1 y 6

```
>> fix(6*rand)+1
```

6*rand produce un valor entre **0 y 5.9999…**
fix(6*rand) produce un resultado entero entre **0 y 5**
fix(6*rand)+1 produce un resultado entero entre **1 y 6**

Ejemplo. Lista de números aleatorios de una cifra

n: cantidad de números
x: número aleatorio de una cifra

```
%Lista de números aleatorios
n=input('Cuantos números ');
for i=1:n
   x=fix(10*rand);
   disp(x);
end
```

Ejemplo. Simule n lanzamientos de un dado. Muestre cuantas veces se obtuvo el 3.

n: dato
i: conteo de lanzamientos
x: cada número aleatorio
c: cantidad de veces que se obtuvo el 3

```
%Simular lanzamientos de un dado
n=input('Ingrese el dato ');
c=0;
for i = 1: n
   x=fix(6*rand)+1;
   if x==3
      c=c+1;
   end
end
disp(c);
```

NOTA: pruebe que si **n** es grande, la respuesta debe ser aproximadamente **1/6** de **n**, según la Teoría de la Probabilidad

Ejemplo. Simular n intentos de un juego con un dado, con la siguientes reglas:

6: gana 4 dólares
3: gana 1 dólar
1: siga jugando
2,4 o 5: pierde 2 dólares

n: dato
i: conteo de lanzamientos
x: cada número aleatorio
s: cantidad acumulada de dinero

```
%Simulación de un juego
n=input('cuantos intentos ');
s=0;
for i=1:n
   x=fix(6*rand)+1;
   switch x
      case 6
         s=s+4;
      case 3
         s=s+1;
      case {2, 4, 5}
         s=s-2;
   end
end
disp(s);
```

Observe que si el número de intentos es grande, normalmente se obtendrá un valor negativo. Esto significa que no es una buena idea jugar.

Ejemplo. Simule n lanzamientos de un dardo en un cuadrado de 1 m de lado. Determine cuantos intentos cayeron dentro de un círculo inscrito en el cuadrado.

n: dato
i: conteo de lanzamientos
x,y : corrdenadas aleatorias de cada lanzamiento
c: cantidad de veces que están dentro del círculo de radio 1

```
%Puntos aleatorios dentro de un círculo
n=input('¿Cuántos intentos? ');
c=0;
for i=1:n
   x=rand;
   y=rand;
   if x^2 + y^2 <= 1
      c=c+1;
   end
end
disp('Dentro del círculo');
disp(c);
```

Salida

```
>> puntos
¿Cuántos intentos? 100
Dentro del círculo
   78
```

Ejemplo. Dado un conjunto de enteros numerados 1 a n, elegir una muestra aleatoria de m números, m≤n

```
%Muestra aleatoria (con repeticiones)
n = input('Ingrese dato de la población ');
m = input('Ingrese tamaño de la muestra ');
for i=1: m
   x=fix(n*rand)+1;
   disp(x);
end
```

Salida

```
Ingrese dato de la población  15
Ingrese tamaño de la muestra  5
10
12
14
12
 3
```

Note que pueden salir valores repetidos. Posteriormente se describirá una técnica para que no se incluyan elementos repetidos

Ejemplo. Dado un número entero positivo n, sume los cuadrados de los enteros entre 1 y n. Verifique si esta suma es múltiplo del dato dado n.

n: número dado
x: cada número entero
s: suma de los cuadrados

```
%Prueba de múltiplos
n=input('Ingrese el valor final ');
s=0;
for x = 1: n
   s=s+x^2;
end
if mod(s, n) == 0
   disp('La suma es múltiplo');
else
   disp('La suma no es múltiplo ');
end
```

Ejemplo. Dado un número entero, encuentre todos sus divisores enteros exactos.
n: dato
x: cada número entero

```
%Encontrar los divisores de un número
n=input('Ingrese el dato ');
for x = 1: n
   if mod(n,x) == 0
      disp(x);
   end
end
```

Salida

```
Ingrese el dato  20
        1
        2
        4
        5
        10
        20
```

Ejemplo. Dado un número entero, cuente sus divisores enteros exactos.
n: dato
x: cada número entero
c: cantidad de divisores

```
%Conteo de los divisores de un número
n=input('Ingrese el dato ');
c=0;
for x = 1: n
   if mod(n,x) == 0
      c=c+1;
   end
end
disp(c);
```

Ejemplo. Dado un número entero, encuentre cuantos divisores enteros exactos tiene y luego determine si es primo

```
%Determinar si un número es primo
n=input('Ingrese el dato ');
c=0;
for x = 1: n
   if mod(n,x) == 0
      c=c+1;
   end
end
if c > 2
disp('No es primo ');
else
   disp('Si es primo ');
end
```

Salida

```
Ingrese el dato  1307
Si es primo
```

4.6.3 Ciclos anidados

Algunos algoritmos requieren ciclos dentro de otros ciclos

Ejemplo. Liste todas las parejas de números con valores entre 1 y 5

a,b: números

```
%Parejas de números
for a = 1: 3
   for b = 1: 3
      disp([a, b]);
   end
end
```

Salida

```
1   1
1   2
1   3
2   1
2   2
2   3
3   1
3   2
3   3
```

El ciclo interno se realiza completamente para cada valor de la variable del ciclo externo. En el ejemplo anterior en total se realizan 3x3 = 9 ciclos

Ejemplo. Liste todas las parejas de números con valores entre 1 y 5 pero sin que hayan parejas repetidas

a,b: números

```
%Parejas de números sin repeticiones
for a = 1: 3
   for b = a: 3
      disp([a, b]);
   end
end
```

Salida

```
1   1
1   2
1   3
2   2
2   3
3   3
```

Ejemplo. Liste todos los números de dos cifras que se pueden hacer con los dígitos del 1 al 3

a, b: dígitos entre 1 y 3
n: número

```
for a = 1: 3
   for b = 1: 3
      n = 10*a + b;
      disp(n);
   end
end
```

Salida

```
11
12
13
21
22
23
31
32
33
```

Ejemplo. Liste todas las ternas (a, b, c) de números enteros entre 1 y 20 que cumplen la propiedad pitagórica: $a^2 + b^2 = c^2$

a, b, c: números entre 1 y 20

```
%Ternas pitagóricas
for a = 1: 20
   for b = 1: 20
         for c = 1: 20
            if a^2 + b^2 == c^2
                  disp([a, b, c]);
            end
      end
   end
end
```

Salida

```
 3    4    5
 4    3    5
 5   12   13
 6    8   10
 8    6   10
 8   15   17
 9   12   15
12    5   13
12    9   15
12   16   20
15    8   17
16   12   20
```

Ejemplo. Liste todas las ternas (a, b, c) de números enteros entre 1 y 20 que cumplen la propiedad pitagórica: $a^2 + b^2 = c^2$ pero sin ternas repetidas

a, b, c: números entre 1 y 20

```
%Ternas pitagóricas sin repeticiones
for a = 1: 20
   for b = a: 20
         for c = b: 20
            if a^2 + b^2 == c^2
                  disp([a, b, c]);
            end
      end
   end
end
```

Salida

```
 3   4   5
 5  12  13
 6   8  10
 8  15  17
 9  12  15
12  16  20
```

Pregunta: ¿Cuantas repeticiones se realizan en total en cada uno de los dos últimos algoritmos?

Nota. Entre dos algoritmos que resuelven el mismo problema, aquel con menor número de ciclos es más eficiente. Este es un criterio para elegir algoritmos

Ejemplo. Determine si la proposición $(a \wedge b) \Rightarrow (a \vee c)$ es una tautología

Se puede usar una expresión equivalente: $\neg((a \wedge b)) \vee (a \vee c)$

```
for a=0:1
   for b=0:1
      for c=0:1
         p=~((a&b))|(a|c);
         disp([a,b,c,p]);
      end
   end
end
```

Salida

```
0   0   0   1
0   0   1   1
0   1   0   1
0   1   1   1
1   0   0   1
1   0   1   1
1   1   0   1
1   1   1   1
```

El resultado muestra que es una tautología. Se puede mostrar este mensaje incluyendo un conteo dentro de los ciclos.

4.6.4 La instrucción break

Se utiliza para salir de una repetición sin completar todos los ciclos

Ejemplo. Simular lanzamientos de un dado. Determinar la cantidad de lanzamientos hasta que salga el 5.

```
n=input('Cantidad máxima de lanzamientos ');
for i=1:n
   x=fix(rand*6)+1;
   disp(x)
   if x==5
      disp('Cantidad de lanzamientos hasta que salió el 5');
      disp(i);
      break;
   end
end
```

Salida

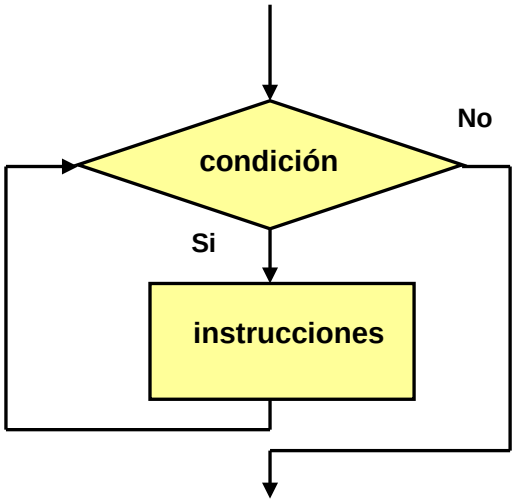
```
Cantidad máxima de lanzamientos  200
    4
    3
    6
    4
    2
    1
    5
Cantidad de lanzamientos hasta que salió el 5
    7
```

Esta solución no luce muy natural pues la salida normalmente debería realizarse cuando se agotan los valores de la lista **for**. La instrucción **break** permite salir del ciclo antes de finalizar el conteo de repeticiones. Adicionalmente, no está garantizado que se obtenga el número 5 en la cantidad de lanzamientos especificada.

Hay una manera más natural de realizar estos algoritmos: la instrucción de repetición **while**

4.6.5 El ciclo WHILE

Representación gráfica



El ciclo WHILE en MATLAB

```
while condición
    instrucciones
end
```

Ejemplo. Simular lanzamientos de un dado. Muestre el resultado en cada intento hasta que salga el 5.

x: resultado del dado en cada lanzamiento. Inicialmente un valor nulo para entrar al ciclo

```
%Simular lanzamientos de un dado hasta que sale un número
x=0;
while x~=5
   x=fix(rand*6)+1;
   disp(x)
end
```

Salida

```
1
3
6
4
2
5
```

En el ejemplo se muestra que el uso de la estructura **while** es natural cuando no se conoce la cantidad de repeticiones que deben realizarse hasta salir del ciclo.

Si es de interés concocer el número de repeticiones realizadas se debe incluir una variable para realizar el conteo como se muetra en el siguiente ejemplo.

Ejemplo. Simular lanzamientos de un dado. Determinar la cantidad de lanzamientos hasta que salga el 5.

x: resultado del dado en cada lanzamiento. Inicialmente un valor nulo para entrar al ciclo
c: conteo de repeticiones

```
%Conteo de lanzamientos de un dado
c=0;
x=0;
while x~=5
   x=fix(rand*6)+1;
   disp(x)
   c=c+1
end
disp('Cantidad de lanzamientos hasta que salió el 5');
disp(c);
```

Salida

```
     2
     4
     2
     6
     1
     4
     5
  Cantidad de lanzamientos hasta que salió el 5
     7
```

Compare esta solución con la propuesta usando **for** y **break**. Esta solución es más natural

Todos los algoritmos de repetición pueden ser descritos con la instrucción **while**, pero si la secuencia de valores es un conteo simple, normalmente es más claro usar la instrucción **for**

Ejemplo. Suma de los cuadrados de los primero n números naturales

Con la instrucción for:

```
%Suma de cuadrados con FOR
n=input('Ingrese el valor final ');
s=0;
for  i = 1: n
   c=i^2;
   s=s+c:
end
disp('La suma es');
disp(s);
```

Salida

```
Ingrese el valor final  5
La suma es
   55
```

Ejemplo. Suma de los cuadrados de los primero n números naturales

Con la instrucción while

```
%Suma de cuadrados con WHILE
n=input('Ingrese el valor final ');
s=0;
i=1;
while i <= n
   c=i^2;
   s=s+c;
   i=i+1;
end
disp('La suma es');
disp(s);
```

Salida

```
Ingrese el valor final  5

La suma es
  55
```

La instrucción **for** luce más adecuada para este ejemplo, pero en los siguientes ejemplos, es mucho más natural usar la instrucción **while**

Ejemplo. Dado una cantidad inicial de bacterias, determine en que día excede al máximo si cada día se duplica la cantidad

x: cantidad inicial
m: valor máximo
d: día

```
%Crecimiento de bacterias
x=input('Ingrese la cantidad inicial ');
m=input('Ingrese la cantidad máxima ');
d=0;
while x < m
   x=2*x;
   d=d+1;
end
disp('Dia ');
disp(d);
```

Ejemplo. Dado un número entero, genere la secuencia con la siguiente regla. Esta secuencia se denomina de Ulam. Finalmente se llega al número 1

$$x = \begin{cases} x/2, & x \text{ par} \\ 3x+1, & x \text{ impar} \end{cases}$$

x: número dado

```
%Secuencia de Ulam
x=input('Ingrese el dato ');
while x ~= 1
   if mod(x,2) == 0
      x=x/2;
   else
      x=3*x+1;
   end
   disp(x);
end
```

Ejemplo. Dado un entero positivo encuentre sus factores primos

Para probar: Si n es 72, sus factores primos son: 2, 2, 2, 3, 3 pues 2x2x2x3x3 = 72

n: dato
p: cada factor primo

```
%Factores primos de un número entero
n = input('Ingrese el dato');
p = 2;
while f<=n
   while mod(n, p) == 0
         disp(p);
         n=fix(n/p);
   end
   p=p+1;
end
```

4.6.6 Ejercicios de programación con ciclos

1.- Calcule el promedio, el menor valor y el mayor valor de los pesos de **n** paquetes en una bodega. Estos datos ingresan uno a la vez dentro de un ciclo. **n** es un dato ingresado al inicio.

2.- Clasifique los pesos de los **n** objetos de una bodega en tres grupos: menor a 10 Kg., entre 10 y 20 Kg., mas de 20 Kg. Los datos ingresan uno a la vez en un ciclo.

3.- Dado un número par **n**, sume los primeros **n** números impares:
Ejemplo. **n=4: s = 1+3+5+7 = 16**
Verifique si esta **s** suma es igual a $\mathbf{n^2}$ y muestre un mensaje

4.- Dado dos números enteros **a, b**, determine su máximo común divisor **m**.
Ejemplo: **a** = 36, **b** = 45 entonces **m** = 9

5.- Dado un número entero positivo **n**, descompóngalo en sus factores primos

Ejemplo: **n** = 72, factores primos: **2, 2, 3, 3, 3**

6.- Lea los votos de **n** personas. Cada voto es un número **1, 2, o 3** correspondiente a tres candidatos. Si el dato es 0 es un voto en blanco. Si es otro número es un voto nulo. Determine el total de votos de cada candidato y el total de votos blancos y nulos.

7.- Lea las coordenadas de **u, v** de la ubicación de una fábrica y las coordenada **x, y** de **n** sitios de distribución. Encuentre cual es la distancia del sitio más alejado de la fábrica

8.- Encuentre el mayor valor de la función **f(x)=sen(x)+ln(x)**, para los valores:
x=1.0, 1.1, 1.2, 1.3, ..., 4

9.- Lea las coordenadas de n puntos en un plano. Calcule la suma de las distancias de los puntos en un recorrido consecutivo del punto 1 al punto n.

10.- Determine la suma de los términos de la serie $\mathbf{1^3 + 3^3 + 5^3 + ... + n^3}$ en donde **n** es un dato

11.- Determine la suma de los n primeros números de la serie:
1, 1, 2, 3, 5, 8, 13, 21, ...
en la cual cada término, a partir del tercero, se obtiene sumando los dos términos anteriores

12.- Determine la suma de los números $\mathbf{2^1 + 2^2 + 2^3 + 2^4 + \ldots + 2^{64}}$

13.- Una empresa compra una máquina en $20000 pagando cuotas anuales durante cinco años. La siguiente fórmula relaciona el costo de la máquina P, el pago anual A, el número de años n y el interés anual r:

$$A = P\frac{r(1+r)^n}{(1+r)^n - 1}$$

Escriba un programa que permita calcular el valor de P para valores de
r = 0.01,0.02, 0.03, …, 0.1

14.- Escriba un programa para controlar la cantidad de contenedores en un patio. Ingrese como dato la cantidad inicial y ofrezca las siguientes opciones:

1. Salida de contenedores
2. Llegada de contenedores
3. Cantidad actual de contenedores
4. Terminar el control

En cada repetición el operador elige la opción ingresando el número y la cantidad de contenedores.

15.- En un supermercado se hace una promoción, mediante la cual el cliente obtiene un descuento dependiendo de un número de una cifra que se escoge al azar. Si el numero

escogido es menor que 7 el descuento es del 5% sobre el total de la compra, si es mayor o igual a 7 el descuento es del 10%. Obtener cuanto dinero se le descuenta.

16.- Programa para simular la extracción de n bolas de una caja que contiene m bolas numeradas del 1 al m. Cada vez que se saca la bola se muestra el número y se la devuelve a la caja, por lo tanto pueden salir bolas repetidas.

17.- Escriba un programa que genere un número aleatorio con un valor entre 1 y 100 y que sea un número primo.

18.- Escriba un programa que muestre dos números aleatorios con valores enteros entre 1 y 100 tales que la suma sea un número primo.

19.- Lea un número par. Encuentre dos números al azar tales que la suma sea igual al dato dado.

20.- Simule el siguiente juego entre tres ranas. Las ranas están al inicio de una pista de 20 m. En turnos cada rana realiza un intento. El intento es una acción aleatoria que puede ser: a) No se mueve, b) Salta 1 m, c) Salta 2 m. Determina cual de las tres ranas llaga primero a la meta.

21.- Realice la simulación de n intentos de lanzamientos de un dado con las siguientes reglas: si sale 6 gana $5. Si sale 1 gana $1. Si sale 2, 3, 4 o 5 pierde $2. Determine la cantidad acumulada al final del juego

22. En el juego del Tarot, para averiguar el número que le corresponde a una persona se suman todos los dígitos de la fecha de nacimiento hasta obtener un solo dígito

Ejemplo: Fecha de Nacimiento: 28 nov. 1989

28 + 11 + 1989 = 2028 --> 2 + 0 + 2 + 8 = 12 --> 1 + 2 = 3

Entonces el número buscado es 3

Lea tres números: dia, mes, año y muestre el número del Tarot correspondiente

5 Vectores en MATLAB

5.1 Definición

Los vectores son dispositivos para representar y manejar variables que pueden tener varios componentes

Definición de un vector

nombre = [componentes]

Notación para los componentes de un vector

Notación Matemática: $\mathbf{x_i}$
Notación MATLAB: **x(i)**

x es el nombre del vector
i es el número de la celda (numeradas en forma natural desde **1**)

Crear un vector fila de tres componentes

```
>> x=[5, 7, 4];
>> x
x =
  5    7    4
```

Los componentes pueden separarse con espacios o con comas. Si se desea crear un vector columna, los elementos deben separarse con punto y coma.

Crear un vector columna de tres componentes

```
>> t=[6; 9; 2];
>> t
 t  =
  6
  9
  2
```

Los vectores pueden manejarse dinámicamente:

Agregar un nuevo elemento al final del vector **x** anterior:

```
>> x=[x, 8];
>> x
  x =
    5    7    4    8
```

Agregar un nuevo elemento al inicio del vector **x** anterior:

```
>> x=[6, x];
>> x
  x =
    6   5    7    4    8
```

Eliminar el tercer elemento del vector **x**

```
>> x(3)= [ ];
>> x
  x =
     6   5    4    8
```

Eliminar el vector completo

```
>> x= [ ];
>> x
    x = [ ]
```

También se puede crear un vector asignando valores a sus componentes:

Crear un vector fila de tres componentes

```
>> x(1)=5;
>> x(2)=7;
>> x(3)=4;
>> x
x =
   5    7    4
```

El manejo individual de los componentes requiere el uso de un índice

Crear un vector fila

```
>> x=[6, 7, 4, 8, 3 ];
>> x
x =
  6    7    4    8    3
```

Mostrar el tercer componente

```
>> x(3)
ans =
  4
```

Sustituir el cuarto componente por el 9

```
>> x(4)=9
x =
   6    7    4    9    3
```

Mostrar los componentes 2, 3 y 4

```
>> x(2:4)
ans =
   7    4    9
```

Insertar en la posición 3 el valor 8

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

5.2 Algunas funciones de MATLAB para manejo de vectores

Longitud de un vector

```
>> v=[2 4 7 3 5 7 8 6];
>> n=length(v)
n =
    8
```

Suma de los componentes de un vector

```
>> s=sum(v)
s =
   42
```

Máximo valor de los componentes

```
>> m=max(v)
m =
    8
```

Máximo valor de los componentes y su posición

```
>> [m,p]=max(v)
m =
    8
p =
    7
```

Media o promedio aritmético

```
>> p=mean(v)
p =
   5.2500
```

Determinar si algún elemento pertenece al vector

```
>> v=[2 4 7 3 5 7 8 6];
>> e=ismember(8,v)
e =
    1
>> e=ismember(9,v)
e =
    0
```

Determinar adicionalmente la posición del elemento en el vector

```
>> [e,p]=ismember(8,v)
e =
    1
p =
    7
```

5.3 Algoritmos con vectores

Nota para pruebas con vectores. Al realizar pruebas con vectores de diferente longitud conviene borrar de la memoria el vector de la ejecución anterior pues el vector aún mantiene los elementos de la ejecución anterior. Se sugiere incorporar en el programa o en la ventana de comandos una instrucción para iniciar el vector en cada prueba insertando el comando **clear** con el nombre del vector, o iniciándolo como un vector vacío definiéndolo con la notación **[ ]**

Ejemplo. **clear x;** o **x=[ ];**

Todas las variables de los programas son visibles desde fuera del programa, es decir que están disponibles y pueden usarse desde la ventana de comandos con el nombre con que fueron creadas.

5.3.1 Ingreso de datos de un vector a un programa

Se pueden ingresar los datos individualmente y agregarlos al vector (agregar cada dato a la derecha). (Previamente requiere conocer cuantos datos se leerán)

```
n=input('cuantos datos? ');
v=[ ];
for i=1:n
   x=input('ingrese dato ');
   v=[v, x];
end
```

También se pueden agregar los datos hacia la izquierda

```
n=input('cuantos datos? ');
v=[ ];
for i=1:n
   x=input('ingrese dato ');
   v=[x, v];
end
```

La manera tradicional es el ingreso de cada dato individualmente al vector usando un índice. (Previamente requiere conocer cuantos datos se leerán)

```
n=input('cuantos datos? ');
for i=1:n
   x=input('ingrese dato ');
   v(i)=x;
end
```

La manera más simple es ingresar el vector completo al programa.
La función **length** detecta la cantiadad de datos que ingresaron al vector

```
n=input('ingrese el vector ');
n=length(v);
```

5.3.2 Asignación de valores aleatorios a un vector dentro de un programa

Asignación individual

Ejemplo. Crear un vector aleatorio agregando cada dato al vector (números de un dígito)

```
n=input('cuantos números? ');
v=[ ];
for i=1:n
   x=fix(rand*10);
   v=[v, x];
end
```

También se puede crear el vector asignando los valores mediante un índice

```
n=input('cuantos números? ');
for i=1:n
   v(i)=fix(rand*10);
end
```

Se puede asignar el vector aleatorio completo en una instrucción

```
n=input('cuantos números? ');
v=fix(rand*10(1,n));
```

Se ha generado un vector de 1 fila y n columnas con números aleatorios de una cifra

Ejemplo. Dado un vector, sume sus componentes.

x: vector
n: número de componentes
s: suma de los componentes

```
x=input('Ingrese vector ');
n=length(x);
s=0;
for i = 1: n
   s=s+x(i);
end
disp('La suma es');
disp(s);
```

Salida

```
Ingrese vector  [7  8  9  12  6]
La suma es
        42
```

NOTA: Este ejercicio emula la función **sum** existente en MATLAB:

```
>> x=[7  8   9  12  6];
>> s=sum(x)
        42
```

Ejemplo. Dado un vector, sume sus componentes con valor impar.

x: vector
n: número de componentes
s: suma de los componentes

```
%Suma de componentes con valor impar de un vector
x=input('Ingrese vector ');
n=length(x);
s=0;
for i = 1: n
   if mod(x(i), 2) ~= 0
      s=s+x(i);
   end
end
disp(s);
```

Salida
Ingrese vector [7 8 9 12 6]
16

Ejemplo. Dado un vector, encuentre el mayor valor.

x: **vector**
n: **número de componentes**
m: **el mayor valor**

```
x=input('Ingrese vector ');
n=length(x);
m=x(1);
for i = 2: n
   if x(i) > m
      m = x(i);
   end
end
disp(m);
```

NOTA: Este ejercicio emula la función max de MATLAB:

```
>> x=[7  8   9  12  6];
>> s=max(x)
          12
```

Ejemplo. Dado un vector, muestre sus componentes en orden opuesto.

x: vector dato
y: vector resultado
n: número de componentes

```
%Invertir un vector fila
x=input('Ingrese vector ');
n=length(x);
j=n;
for i = 1:n
    y(i)=x(j);
    j=j-1;
end
disp(y);
```

Ejemplo. Otra solución para invertir un vector

```
%Invertir un vector
x=input('Ingrese vector ');
n=length(x);
y=[ ];
for i = 1:n
    y=[x(i),y];
end
disp(y);
```

Ejemplo. Generar un vector con números aleatorios de dos cifras. Calcular el promedio y mostrar los elementos mayores al promedio

n: número de componentes
x: vector
p: promedio

```
n=input('Cantidad de elementos ');
for i = 1: n
   x(i)=fix(rand*100);
end
disp(x);
p=mean(x);
disp(p);
for i = 1: n
   if x(i)>p
     disp(x(i));
   end
end
```

```
>> Prueba
    Cantidad de elementos  8
   81    90    12    91    63     9    27    54
   53.3750
   81
   90
   91
   63
   54
```

vector
promedio
elementos mayores al promedio

Ejemplo. Almacene en un vector los números de la secuencia de Ulam y muestre la secuencia comenzando desde el final

x: valor inicial para la secuencia
u: vector con los números de la secuencia
i: indice

```
x=input('Valor inicial para la secuencia ');
n=0;
while x>1
    if mod(x, 2) == 0
        x=x/2;
    else
        x=3*x+1;
    end
 n = n + 1;
 u(n) = x;
end
for i=n:-1:1
  disp(u(i));
end
```

Salida

Valor inicial para la secuencia 10
1
2
4
8
16
5

Otra solución para el ejemplo anterior (vector con números de la secuencia de Ulam)

En esta solución el vector se construye dinámicamente

x: valor inicial para la secuencia
u: vector con los números de la secuencia

```
x=input('Valor inicial para la secuencia ');
u=[ ];
while x>1
    if mod(x, 2) == 0
        x=x/2;
    else
        x=3*x+1;
    end
 u=[u, x];
end
disp(u);
```

Salida

Valor inicial para la secuencia 10
5 16 8 4 2 1

Ejemplo. Dado un número entero, encuentre los dígitos de su equivalente en el sistema binario.

x: dato
b: vector que contiene los dígitos en el sistema binario.

```
x=input('Ingrese un numero ');
b=[ ];
while x>0
   d=mod(x,2);
   x=fix(x/2);
   b = [d, b];
end
disp(b);
```

Salida

Ingrese un numero 23
1 0 1 1 1

Nota: 23 en el sistema decimal, es equivalente a 10111 en el sistema binario

Ejemplo. Construya un vector con **n** números primos

n: cantidad de números primos
x: números naturales
p: vector con los números primos

```
n=input('¿Cuantos números primos? ');
x=1;
p = [ ];
while length(p) < n
        c=0;
        for d=1: x
                if mod(x, d) == 0
                        c=c+1;
                end
        end
        if c<=2
                p = [p, x];
        end
        x=x+1;
end
disp(p);
```

Salida

¿Cuantos números primos? 8

1 2 3 5 7 11 13 17 19

Ejemplo. Simule **n** lanzamientos de un dado. Muestre la cantidad de veces que sale cada número.

n: dato
d: número obtenido para el dado en cada lanzamiento
c: vector con la cantidad de resultados de cada número del dado

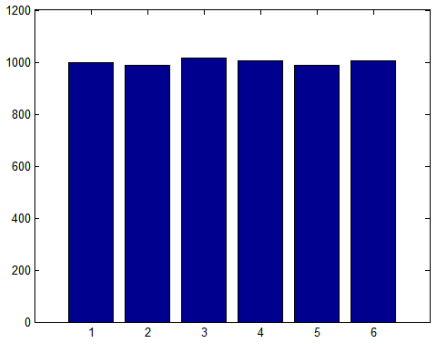
```
%Conteo de resultados de un dado
n=input('¿cuantas pruebas? ');
c=[0 0 0 0 0 0];
for i=1:n
    d=fix(rand*6)+1;
    switch d
       case 1, c(1) = c(1)+1;
       case 2, c(2) = c(2)+1;
       case 3, c(3) = c(3)+1;
       case 4, c(4) = c(4)+1;
       case 5, c(5) = c(5)+1;
       case 6, c(6) = c(6)+1;
    end
end
disp(c);
```

Salida

```
>> dado
¿cuantas pruebas? 6000
     999     988     1016     1007     986     1004
```

Obtenga un gráfico de los resultados

```
>> bar(c)
```



Se observa que la cantidad de resultados es aproximadamente igual, como se esperaba

Ejemplo. Simule **n** lanzamientos de un dado. Muestre la cantidad de veces que sale cada número.

Una solución más compacta para el ejemplo anterior

n: dato
d: número obtenido para el dado en cada lanzamiento
c: vector con la cantidad de resultados de cada número del dado

```
%Conteo de resultados de un dado
n=input('¿cuantas pruebas? ');
c=[0 0 0 0 0 0];
for i=1:n
    d=fix(rand*6)+1;
    c(d)=c(d)+1;
end
disp(c);
```

Ejemplo. Búsqueda en un vector. Escriba un programa que lea desde el teclado un vector y un número y determine si el valor se encuentra en el vector.

v: vector con datos
x: valor que se desea buscar en el vector
e: resultado de la búsqueda (valor lógico)

Soluciones incompletas

```
v=input('ingrese vector ');
n=length(v);
x=input('ingrese dato para buscar ');
for i=1:n
   if x==v(i)
      disp('si');
   else
      disp('no');
   end
end
```

```
>> v=[7,9,6,8,9];
>> pr
ingrese vector v
ingrese dato para buscar 9
no
si
no
no
si
```

```
v=input('ingrese vector ');
n=length(v);
x=input('ingrese dato para buscar ');
for i=1:n
   if x==v(i)
      e=1;
   else
      e=0;
   end
end
if e==1
   disp('si')
else
   disp('no');
end
```

```
>> v=[7,9,6,8,9];
>> pr
ingrese vector v
ingrese dato para buscar 8
no
```

Solución correcta

```
v=input('ingrese vector ');
n=length(v);
x=input('ingrese dato para buscar ');
e=0;
for i=1:n
   if x==v(i)
      e=1;
   end
end
if e==1
   disp('si')
else
   disp('no');
end
```

```
>> v=[7,9,6,8,9];
>> pr
ingrese vector v
ingrese dato para buscar 8
si
```

Ejemplo. Determinar si un elemento está en el vector y su posición

v: vector con datos
x: valor que se desea buscar en el vector
e: resultado de la búsqueda (valor lógico)
p: posición del valor x en el vector v

```
v=input('ingrese el vector de datos ');
x=input('ingrese el número para buscar ');
n=length(v);
e=0;
p=0;
for i=1:n
        if x == v(i)
           e=1
           p=i;
        end
end
if e==0
        disp('el valor no esta en el vector');
else
        disp('el valor está almacenado en la posición');
        disp(p);
end
```

```
>> v=[7,9,6,8,9];
>> pr
ingrese el vector de datos   v
ingrese el número para buscar    8
el valor está almacenado en la posición
        4
```

NOTA: Se puede resolver directamente con la función **ismember** de MATLAB:

```
>> v=[7,9,6,8,9];
>> e=ismember(8,v)
e =                                   el valor 1 indica que 8 está incluido en v
   1
```

```
>> [e, p]=ismember(8,v)
e =
    1
p =
    4
```

también se puede conocer su posición en v

Ejemplo. Obtener una muestra de tamaño **m** de un conjunto de tamaño **n** sin repeticiones

```
n=input('tamaño población ');
m=input('tamaño muestra ');
v=[ ];
while length(v)<m
   x=fix(rand*n)+1;
   if ismember(x,v) == 0
      v=[v, x];
   end
end
disp(v)
```

Ejemplo. Generar una tabla de 15 números con valores entre 1 y 25

```
t=[ ];
while length(t)<15
        x=fix(rand*25)+1;
        if ismember(x,t)==0
                t=[t,x];
        end
end
disp(t);
```

Ejemplo. Escriba un programa que lea desde el teclado dos vectores y determine la cantidad de elementos comunes entre ambos vectores.

a, b: vectores con datos
c: conteo de datos comunes

```
a=input('primer vector ');
b=input('segundo vector ');
c=0;
for i=1: length(a)
   if ismember(a(i),b)==1
     c=c+1;
   end
end
disp('cantidad elementos comunes');
disp(c);
```

Ejemplo. Escriba un programa que lea desde el teclado dos vectores y determine cuales son los elementos comunes entre ambos vectores.

a, b: vectores con datos
x: vector con elementos comunes

```
a=input('primer vector ');
b=input('segundo vector ');
x=[ ];
for i=1: length(a)
   if ismember(a(i),b)==1
     x=[x, a(i)];
   end
end
disp('elementos comunes');
disp(x);
```

Ejemplo. Colocar el mayor valor de un vector en la última posición

x: vector
n: número de componentes
El algoritmo compara cada dato con el último elemento, cada vez que es mas grande, los intercambia.

```
x=input('Ingrese vector ');
n=length(x);
for j = 1: n-1
   if x(j) > x(n)
      t = x(j);
      x(j) = x(n);
      x(n) = t;
   end
end
disp(m);
```

Ejemplo. Ordenar los datos de un vector

Lea un vector y ordene los valores en forma creciente.
x: vector con los datos, su contenido es modificado para ordenarlo
Este algoritmo extiende la idea del algoritmo anterior

```
x=input('Ingrese un vector ');
n=length(x);
for k=n:-1:2
    for j=1:k-1
          if x(j) > x(k)
             t=x(j);
    x(j)=x(k);
       x(k)=t;
          end
    end
end
disp(x)
```

```
>> orden
Ingrese un vector [10 3 8 12 5 7]

    3     5     7     8    10    12
```

Ejemplo. Búsqueda en un vector almacenado en el disco

Primero almacene en el disco un vector con datos. Escriba el vector en la ventana de comandos y guarde en el disco en un archivo con algún nombre. Ej. 'data'

```
>> x=[23, 45, 38, 27, 56, 72, 34, 27, 44];
>> save 'data'  x
```

Escriba un programa que lea desde el teclado un número. Reciba el vector recuperado del disco y determine si el valor se encuentra en el vector.
x: vector con datos recuperado del archivo 'data' del disco
t: valor que se desea buscar en el vector
p: posición del valor t en el vector x

```
x=input('ingrese el vector de datos ');
t=input('ingrese el número para buscar ');
n=length(x);
p=0;
for i=1:n
    if t == x(i)
       p=i;
   end
end
if p==0
    disp('el valor no está almacenado');
else
   disp('el valor está almacenado en la posición');
   disp(p);
end
```

```
>> load 'data'  x
>> buscar
    ingrese el vector de datos  x
    ingrese el número para buscar  34

    el valor está almacenado en la posición
       7
```

Primero se recupera el vector desde el disco

NOTA
El vector **X** quedará almacenado permanentemente en el disco en un archivo con el nombre 'data' con extensión **.mat** Si desea borrarlo puede hacerlo desde fuera de MATLAB o desde la ventana de comandos de MATLAB con el comando **delete**.

```
>> delete data.mat
```

5.4 Ejercicios con vectores

1.- Lea una lista de los pesos de las cajas en un contenedor. Determine cuantos exceden al peso promedio del grupo.

2.- Lea una lista de los pesos de los objetos en una bodega. Determine cual es el rango de los pesos de los objetos.

3.- Escriba un programa para controlar el uso de los camiones de una empresa. Cada camión tiene un código. Ingrese como dato inicial la lista de los códigos de los camiones. Programe una aplicación con las siguientes opciones:

1. Salida de un camión
2. Devolución de un camión
3. Disponibilidad de un camión
4. Terminar

El operador elige la opción ingresando el número.

4.- Lea los códigos de las cajas de un contenedor. Determine si hay códigos repetidos

5.- Una bodega contiene n paquetes numerados en forma natural. Para una inspección se debe tomar una muestra aleatoria del 10% de los paquetes. Escriba un programa para elegir la muestra.

6.- Para la inspección de los **m** paquetes de una bodega se han elegido a **m** inspectores. Realice aleatoriamente la asignación de tal manera que cada inspector se le asigne la revisión de un solo paquete.

7.- Se tiene la lista de los códigos de las cajas distribuidas en dos bodegas y una lista de los códigos de las cajas que se requieren para una inspección. Determine cuantas de estas cajas están en cada bodega.

8. Almacene en un vector **U** las abscisas y en un vector **V** las ordenadas de un conjunto de n puntos en un plano. Determine cual es el punto más alejado del origen. Use la fórmula de la distancia.

9.- Cálculo del área de un polígono arbitrario, dadas las coordenadas de sus vértices
$(x_1, y_1)(x_2, y_2), \ldots, (x_n, y_n)$

$\text{Area} = [(x_1+x_2)(y_1-y_2) + (x_2+x_3)(y_2-y_3) + \ldots + (x_{n-1}+x_{n-2})(y_{n-1}-y_{n-2}) + (x_n+x_1)(y_n-y_1)]/2$

Escriba un programa para leer los valores de x en un vector y los valores de y en otro vector. Calcule u muestre el valor del área con la fórmula anterior

10. En un proceso electoral se tienen anotados los **n** votos para aprobar una moción. Cada voto tiene el número de identificación del elector (números enteros del **1** al **n**) y un número que representa su decisión: 1 si es a favor, 0 si es en contra, cualquier otro número es nulo. Escriba un programa que lea los **n** datos conteniendo el número del elector (no suponga que están ordenados) y su voto. Coloque los números de identificación en tres listas: votantes a favor, votantes en contra y votantes nulos. Finalmente busque y muestre si hay números de identificación de electores que están en más de una lista.

6 Cadenas de caracteres

6.1 Algunos comandos para cadenas de caracteres

Una cadena de caracteres se las expresa con la notación de apóstrofes:

Asignación directa de la cadena de caracteres

```
>> x='problema';
```

Subcadena

```
>> t = x(2:6)
t =
    roble
```

Agregar y eliminar caracteres

```
>> x=[x,'s']
x =
problemas
>> x(8)=''
x =
problems
```

Insertar caracteres

```
>> x=[x(1:3),'x',x(4:9)]
x =
   proxblems
```

Cadena nula

```
>> x='';                                o puede usar [ ]
```

Determinar si un carácter se encuentra en una cadena

```
>> x='programa';
>> e=ismember('r',x)
e =
    1
```

También se puede conocer su posición

```
>> [e,p]=ismember('r',x)
e =
     1
p =
    2
```

Entrega la última posición si hay más de una coincidencia

```
>> x='programa';
>> [e,p]=ismember('a',x)
e =
    1
p =
    8
```

Examina la coincidencia y posición de cada carácter si hay más de uno

```
>> [e,p]=ismember('rta',x)
e =
    1    0    1
p =
    5    0    8
```

Determinar si una cadena se encuentra en otra cadena
El resultado es un vector con las posiciones de coincidencia

```
>> x='En esta prueba para esta materia';
>> t='esta';
>> e=strfind(x,t)
e =
    4    21
>> length(e)
ans =
    2                                          Es la cantidad de coincidencias
```

Determinar si dos cadenas son idénticas: el resultado es un valor lógico

```
>> x='abc';
>> y='abc';
>> t=strcmp(x,y)
t =
    1
```

Funciones para convertir entre numérico y cadena de caracteres

```
>> x=234;
>> c=num2str(x)
c
    = 234

>> c='345';
>> n=str2num(c);
n
    = 345
```

LISTAS DE CADENAS DE CARACTERES

```
>> a=['abc';'rst';'xyz']          Las cadenas deben tener igual longitud
a =
abc
rst
xyz
>> a(2,:)
ans =
rst

>> a=char('abcd','rs','uwxyz','tx')      La function char forma listas con cadenas
a =                                       que pueden tener diferente longitud
abcd
rs
uwxyz
tx
>> [n,k]=size(a)
n=
    4                                  Primer componente de size: cantidad de cadenas
k=
    5                                  Segundo componente de size: la mayor longitud
```

Ejemplo. Salida de un título junto a un número. También puede usarse en input

```
for i=1:5
        disp(['dato ', num2str(i)])
end
```

```
dato 1
dato 2
dato 3
dato 4
dato 5
```

Lectura de una cadena

```
x = input('Ingrese una cadena ');
```

El dato debe ser ingresado encerrado entre comillas simples o mediante una variable previamente asignada con una cadena

Se pueden omitir las comillas al ingresar el dato con especificando 's':

```
x = input('Ingrese una cadena ', 's');
```

6.2 Algoritmos con cadenas de caracteres

Ejemplo. Leer una cadena de caracteres y mostrarla con los caracteres puestos en orden inverso

```
x=input('ingrese una frase ');
n=length(x);
y= '';                          % también puede iniciarla con  [ ]
for i=1:n
   y=[x(i),y];                  % insertar carácter a la izquierda de la cadena
end
disp(y)
```

```
>> invertir
ingrese una frase esta es una prueba
abeurp anu se atse
```

```
>> invertir
ingrese una frase abeurp anu se atse
esta es una prueba
```

Ejemplo. Determinar si una palabra es igual invertida

```
x=input('ingrese una frase ');
n=length(x);
y=[ ];
for i=1:n
   y=[x(i),y];
end
e=strcmp(x,y);
if e==1
  disp('si');
else
   disp('no');
end
```

Ejemplos: radar, reconocer, anita lava la tina

Ejemplo. Leer una frase y determinar cuantas palabras contiene

```
x=input('ingrese frase ',’s’);
n=length(x);
c=1;
for i=1:n
   if x(i)==' '
      c=c+1;
   end
end
disp([‘cantidad de palabras ’, num2str(c)]);
```

Ejemplo. Leer una frase y determinar cuántas veces contiene a una palabra

```
x=input('ingrese frase ',’s’);
p=input(‘ingrese palabra’,’s’);
e=strfind(x,p);                    %Vector conteniendo los inicios de coincidencias
n=length(e);
disp(n);
```

Ejemplo. Leer una cadena de texto y eliminar los espacios en blanco

Solución 1. El resultado se coloca en otra variable

```
x=input('ingrese frase ',’s’);
n=length(x);
y=[ ];
for i=1:n
   if x(i)~=' '
      y=[y,x(i)];
   end
end
disp(y);
```

Solución 2. La misma cadena de entrada es modificada

```
x=input('ingrese frase ',’s’);
n=length(x);
i=0;
while i<length(x)                  % No se puede usar FOR debido a que la longitud
   i=i+1;                  % cambia dinámicamente
   if x(i)==' '
      x(i)=[ ];                    % Elimina el carácter. También se puede: x(i) = '';
      i=i-1;                       % Reducir en 1 la longitud de la cadena
   end
end
disp(x);
```

Ejemplo. Leer una frase y mostrarla enmascarada intercambiando parejas consecutivas de caracteres

```
x=input('Ingrese mensaje ',’s’);
y='';
n=length(x);
for i=1:2:n-1
   a=x(i);
   b=x(i+1);
   y=[y,b,a];
end
if mod(n,2)==1
   y=[y,x(n)];                    %si n es impar, agregar el último carácter
end
disp(y);
```

```
>> codificar
Ingrese mensaje  esta es una prueba
seate  snu arpeuab

>> codificar
Ingrese mensaje  seate  snu arpeuab
esta es una prueba
```

Ejemplo. Diseñe un esquema para enmascarar una frase colocando los caracteres alternadamente alrededor del centro.

Codificar el mensaje

```
%Enmascara colocando caracteres alrededor del centro
x=input('Ingrese mensaje ','s');
y=[ ];
if mod(length(x),2)==1
   x=[' ',x];
end
n=length(x);
for i=1:2:n
   a=x(i);
   b=x(i+1);
   y=[b,y,a];
end
disp(y);
```

Decodificar el mensaje

```
%Decodifica el mensaje del algoritmo anterior
x=input('Ingrese mensaje ','s');
y=[ ];
n=length(x);
for i=1:n/2
   a=x(n/2+i);
   b=x(n/2-i+1);
   y=[y,a,b];
end
disp(y);
```

Ejemplo. Lea nombres que pueden tener diferente longitud y forme una lista. Muestre por cada nombre, la cantidad de veces que contiene la letra ‘a’

Note el uso de **char** para crear la lista de nombres de diferente longitud

```
%Lista vertical de nombres
n=input('cantidad de nombres ');
t=input('ingrese primer nombre ','s');
x=char(t);                                  %inicia lista vertical
for i=1:n-1
   t=input('ingrese siguiente nombre ','s');
   x=char(x, t);                  % char crea una lista vertical
end                               % con los siguientes n-1 nombres
for i=1:n
   v=strfind(x(i,:),'a');         % vector de posiciones de coincidencias de 'a' en x
   c=length(v);
disp([i, c]);
end
```

Ejemplo. Lea una lista de nombre, elimine los nombres con 'r'

```
t=input('ingrese lista ');
i=1;
while i<size(t,1)                 %El primer componente de size es la longitud de la lista
    if ismember('r',t(i,:))==1
       t(i,:)='';
    end
    i=i+1;
end
disp(t)
```

```
>> t=char('maria','juan','pedro','luis')
t =
maria
juan
pedro
luis

>> prueba
ingrese lista t
juan
luis
```

7 Matrices en MATLAB

Matriz: Arreglo rectangular de números organizados en filas y columnas
Definición: **[**elementos de fila 1; elementos de fila 2; elementos de fila 3; **]**
Uso: **(**índice de filas, índice de columnas**)**

7.1 Matrices en la ventana de comandos

Definición

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

Manejo de índices

Elementos

```
>> e=a(3,2)
e =
    9
```

Fila

```
>> d=a(2, 1:3)
d =
    4  6  2
```

Columna

```
>> d=a(1:3,2)
d =
    5
    6
    9
```

Notación abreviada

```
>> d=a(2, :)
d =
    4  6  2
```

Submatrices

```
>> t=a(1:2, 2:3)
t =
    5    7
    6    2
```

Agregar fila

```
>> a=[2, 4, 5; 6, 0, 7; -2, 8, 3]
a =
    2    4    5
    6    0    7
   -2    8    3
```

```
>> a(4,:)=[5 7 9]
a =
    2    4    5
    6    0    7
   -2    8    3
    5    7    9
```

Agregar columna

```
>> a(:,4)=[3;5;7;9]
a =
   2   4   5   3
   6   0   7   5
  -2   8   3   7
   5   7   9   9
```

Eliminar columna

```
>> a(:,4)=[ ]
a =
   2   4   5
   6   0   7
  -2   8   3
   5   7   9
```

Eliminar fila

```
>> a(3,:)=[ ]
a =
   2   4   5
   6   0   7
   5   7   9
```

Insertar columna

```
>> b=[3;9;8]
b =
   3
   9
   8
>> a=[a(:,1),b,a(:,2:3)]
a =
   2   3   4   5
   6   9   0   7
   5   8   7   9
```

Matrices especiales

Matriz de ceros

```
>> a=zeros(3,4)
a =
   0   0   0   0
   0   0   0   0
   0   0   0   0
```

Matriz con 1's

```
>> a=ones(3,4)
a =
   1   1   1   1
   1   1   1   1
   1   1   1   1
```

Matriz identidad

```
>> a=eye(4,4)
a =
   1   0   0   0
   0   1   0   0
   0   0   1   0
   0   0   0   1
```

Matriz con números aleatorios

```
>> a=rand(4,3)
a =
   0.9218   0.9355   0.0579
   0.7382   0.9169   0.3529
   0.1763   0.4103   0.8132
   0.4057   0.8936   0.0099

>> a=fix(rand(4,3)*10)
a =
    8    5    7
    5    4    9
    3    6    5
    7    6    8

>> t=fix(rand(5,3)*25+1)
t =
     6    12    24
     3    10    25
     3    20     5
     2    16     4
    11    20    18
```

Funciones especiales para matrices

```
>> a=[2, 4, 5; 6, 0, 7; -2, 8, 3; 9, 2, 8]
a =
    2    4    5
    6    0    7
   -2    8    3
    9    2    8
```

Dimensiones de una matriz

```
>> [n,m]=size(a)
n =
    4
m =
    3
```

Suma de columnas

```
>> c=sum(a)
c =
   15   14   23
```

Suma de filas

```
>> f=sum(a')
f =
   11   13    9   19
```

a' es la transpuesta de **a**

Suma de todos los elementos

```
>> s=sum(f)
s =
   52
```

El mayor valor de cada columna

```
>> max(a)
ans =
    9    8    8
```

El mayor valor de cada columna y su posición de fila

```
>> [z,p]=max(a)
z =
    9    8    8
p =
    4    3    4
```

El mayor valor de cada fila

```
>> e=max(a')
e =
    5    7    8    9
```

El mayor valor de cada fila columna y su posición en la columna

```
>> [z,p]=max(a')
z =
    5    7    8    9
p =
    3    3    2    1
```

Determinar si un elemento se encuentra en una matriz

```
>> a=[2, 4, 5; 6, 0, 8; -2, 8, 3; 9, 2, 7]
a =
    2    4    5
    6    0    8
   -2    8    3
    9    2    7
>> e=ismember(8,a)
e =
    1
```

Determinar si un elemento se encuentra en una matriz y su posición

```
>> [e,p]=ismember(8,a)
e =
    1
p =
    10
```

Es la posición de la última aparición del número, contando por columnas desde el inicio

Determinar si los elementos de un vector se encuentran en una matriz

```
>> e=ismember([5, 1, 7],a)
e =
    1    0    1
>> e=ismember([5, 1, 7, 9],a)
e =
    1    0    1    1
```

Determinar si una fila se encuentra en una matriz

```
>> e=ismember([5, 1, 7],a,'rows')
e =
    0
>> e=ismember([6 0 7],a,'rows')
e =
    1
```

Elementos de la diagonal de una matriz

```
>> a=[2, 4, 5; 6, 9, 7; -2, 8, 3]
a =
    2    4    5
    6    9    7
   -2    8    3
```

```
>> d=diag(a)
d =
    2
    9
    3
```

Construir una matriz diagonal con un vector

```
>> e=diag(d)
e =
    2    0    0
    0    9    0
    0    0    3
```

Matriz triangular superior

```
>> t=triu(a)
t =
    2    4    5
    0    9    7
    0    0    3
```

Matriz triangular inferior

```
>> r=tril(a)
r =
    2    0    0
    6    9    0
   -2    8    3
```

Suma de los elementos de la matriz triangular superior

```
>> s=sum(triu(a))
s =
    2   13   15
```

Ordenamiento de una matriz por columnas

```
>> a=[2, 4, 5; 6, 0, 7; -2, 8, 3; 9, 2, 8]
a =
    2    4    5
    6    0    7
   -2    8    3
    9    2    8

>> b=sort(a)
b =
   -2    0    3
    2    2    5
    6    4    7
    9    8    8
```

Funciones especiales para matrices

Determinante

```
>> b=[5, 3, 2; 0, 8, 1; 2, 5, 4]
b =
    5    3    2
    0    8    1
    2    5    4

>> d=det(b)
d =
   109
```

Inversa

```
>> c=inv(b)
c =
   0.2477   -0.0183   -0.1193
   0.0183    0.1468   -0.0459
  -0.1468   -0.1743    0.3670
```

Producto de una matriz por su inversa

```
>> d=b*c
d =
    1    0    0
    0    1    0
    0    0    1
```

Resolución de un sistema de ecuaciones lineales

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

>> b=[4;5;6]
b =
    4
    5
    6

>> x=inv(a)*b
x =
  -0.7500
   1.4063
  -0.2187
```

Reducción a forma escalonada canónica

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

>> rref(a)
ans =
   1.0000         0         0   -0.7500
        0    1.0000         0    1.4063
        0         0    1.0000   -0.2188
```

Valores propios de una matriz

```
>> c=eig(a)
c =
  -0.8910 + 2.8862i
  -0.8910 - 2.8862i
  15.7820
```

Operaciones con matrices

```
>> a=[2, 4, 5; 6, 0, 7; -2, 8, 3]
a =
    2    4    5
    6    0    7
   -2    8    3
```

```
>> b=[5, 3, 2; 0, 8, 1; 2, 5, 4]
b =
    5    3    2
    0    8    1
    2    5    4

>> c=a+b
c =
    7    7    7
    6    8    8
    0   13    7

>> c=a*b
c =
   20   63   28
   44   53   40
   -4   73   16

>> d=2*c
d =
   40  126   56
   88  106   80
   -8  146   32

>> e=d+3
e =
   43  129   59
   91  109   83
   -5  149   35
```

Transpuesta de una matriz

```
>> a=[2, 4, 5; 6, 0, 7; -2, 8, 3]
a =
    2    4    5
    6    0    7
   -2    8    3

>> u=a'
u =
    2    6   -2
    4    0    8
    5    7    3
```

Producto punto entre matrices

```
>> a=[2, 4, 5; 6, 0, 7; -2, 8, 3]
a =
    2    4    5
    6    0    7
   -2    8    3

>> b=[5, 3, 2; 0, 8, 1; 2, 5, 4]
b =
    5    3    2
    0    8    1
    2    5    4
>> c=a.*b
c =
   10   12   10
    0    0    7
   -4   40   12
```

7.2 Algoritmos con matrices

Inicialmente de desarrollarán programas simulando algunas de las funciones existentes en MATLAB. Posteriormente, se podrá enfrentar el desarrollo de nuevos programas

Ejemplo. Escriba un programa para simular la función **sum** de MATLAB. El resultado es un vector con las sumas de las columnas

```
>> a=[2, 4, 5; 6, 0, 7; 2, 8, 3]
a =
    2    4    5
    6    0    7
    2    8    3
>> s=sum(a)
s =
    10    12    15
```

```
%Simular la función sum de MATLAB
a=input('Ingrese la matriz ');
[n,m]=size(a);
s=[ ];
for j=1:m                          %Para cada columna se realiza la suma de filas
   t=0;
   for i=1:n
      t=t+a(i,j);
   end
   s=[s, t];
end
disp('Suma de columnas');
disp(s);
```

```
>> sumac
Ingrese la matriz  a
Suma de columnas
s =
    10    12    15
```

Ejemplo. Escriba un programa para simular la función **max** de MATLAB. El resultado es un vector con el mayor valor de cada columna

```
>> a=[2, 4, 5; 6, 0, 7; 2, 8, 3]
a =
    2    4    5
    6    0    7
    2    8    3
>> s=max(a)
s =
    6    8    7
```

Algoritmo:

Para cada columna **j=1,2,3,…,m**
- Coloque el primer elemento **a(1,j)** en la variable t
- Compare los siguientes valores **a(i,j), i=2,3,4, …, n** con el valor **t**
- Si es mayor, reemplace al valor de **t**
- Agregue este valor al vector **s**, el cual contendrá el mayor de cada columna

```
%Simular la función max de MATLAB
a=input('Ingrese la matriz ');
[n,m]=size(a);
r=[ ];
for j=1:m
   t=a(1,j);
   for i=2:n
      if a(I,j)>t
         t=a(i,j);
      end
   end
   r=[r, t];
end
disp('El mayor por columnas');
disp(r);
```

```
>> maxc
Ingrese la matriz  a
El mayor por columnas
r =
    10    12    15
```

Ejemplo. Suma de los elementos con valor par de una matriz

```
a=input('Ingrese la matriz ');
[n,m]=size(a);
s=0;
for i=1:n
   for j=1:m
      if mod(a(i,j),2) == 0
         s=s + a(i,j);
      end
   end
end
disp('la suma es')
disp(s);
```

```
>> sumap
ingrese la matriz  a
la suma es 22
```

Ejemplo. Sume los elementos de una matriz triangular inferior

La respuesta se la puede encontrar directamente sumando los elementos de la matriz triangular inferior obtenida con la función **sum(tril( )).** El siguiente programa producirá el mismo resultado:

```
a=input('ingrese matriz ');
[n,m]=size(a);
s=0;
for i=1:n                          %Sumar cada elemento  a(i,j)  para el cual  i≤j
   for j=1:n
      if i<=j
         s=s+a(i,j);
      end
   end
end
disp(s)
```

Otra solución

```
a=input('ingrese matriz ');
[n,m]=size(a);
s=0;
for i=1:n
   for j=1: i                    %Mediante los índice se suman los elementos
       s=s+a(i,j);
   end
end
disp(s)
```

Ejemplo. Convertir una matriz a un vector fila

```
a=input('ingresar matriz ');
[n,m]=size(a);
v=[ ];
for i=1:m
   for j=1:n
      v=[v, a(i,j)];
   end
end
disp(v);
```

Ejemplo. Convertir una vector a una matriz compatible.

```
v=input('ingrese vector ');
n=input('cuantas filas ');
m=input('cuantas columnas ');
if n*m~=length(v)
   disp('Incompatible');
else
   a=[ ];                                   %para eliminar datos anteriores
   k=1;
   for i=1:n
      for j=1:m
         a(i,j)=v(k);
         k=k+1;
      end
   end
   disp(a);
end
```

Ejemplo. Leer el cuadro de goles de un campeonato de fútbol y producir el cuadro de resultados en puntos. Los goles están anotados por filas.

Algoritmo

Ingrese el cuadro de goles en la matriz **g**
Detemine la dimensión **n**
Inicie en ceros la matriz de resultados **r**
Para cada elemento **g(i,j)** del cuadro de goles
si es igual al elemento opuesto en el cuadro **g(j,i),** es un empate
r(i,j)=1
r(j,i)=1
fin
si g(i,j) > g(j,i) El equipo **i** gana al equipo **j**
r(i,j)=3
r(j,i)=0
fin
fin
muestre el cuadro de resultados r

```
%Leer cuadro de goles de un campeonato de fútbol y mostrar cuadro de resultados
g=input('Ingrese el cuadro de goles ');
[n,n]=size(g);
r=zeros(n,n);
for i=1:n
   for j=1:n
      if i~=j
         if g(i,j)==g(j,i)
            r(i,j)=1;
            r(j,i)=1;
         end
         if g(i,j)>g(j,i)
            r(i,j)=3;
            r(j,i)=0;
         end
      end
   end
end
disp('Cuadro de resultados');
disp(r);
```

```
>> g=fix(5*rand(6,6));                    Tabla de goles (aleatoria)
>> g=g-diag(diag(g))                      Para eliminar la diagonal
g =
    0    4    0    3    2    2
    1    0    0    3    3    1
    2    2    0    3    2    0
    0    0    0    0    4    3
    4    2    2    0    0    3
    4    1    2    1    4    0

>> prueba1
Ingrese el cuadro de goles  g
Cuadro de resultados
    0    3    0    3    0    0
    0    0    0    3    3    1
    3    3    0    3    1    0
    0    0    0    0    3    3
    3    0    1    0    0    0
    3    1    3    0    3    0
```

Ejemplo. Leer el cuadro de resultados de un campeonato de fútbol y producir la lista del o los ganadores. Los resultados están tabulados por filas. Este programa es el complemento del programa anterior.

Algoritmo

Ingresar el cuadro de resultados en la matriz **r**
Determinar la dimensión **n**
Sumar las filas y almacenar los resultados en el vector **s**
Asignar a **t** el máximo puntaje
Almacenar en el vector v todas las filas cuyo puntaje coincida con el valr **t**
Mostrar el contenido del vector **v**
(Si hay más de uno, significa que hay empate)

```
%Leer el cuadro de resultados del campeonato de fútbol y mostrar el(o los) ganadores
r=input('Ingrese la tabla de resultados ');
[n,n]=size(r);
s=sum(r');
t=max(s);
v=[ ];
for i=1:n
   if s(i)==t
      v=[v, i];
   end
end
if length(v)==1
  disp('Equipo ganador');
  disp(v);
else
  disp('Empate entre los equipos');
  disp(v);
end
```

Resultado de la ejecución del programa anterior

```
 >> r
r =
    0    3    0    3    0    0
    0    0    0    3    3    1
    3    3    0    3    1    0
    0    0    0    0    3    3
    3    0    1    0    0    0
    3    1    3    0    3    0

>> prueba2
Tabla de resultados  r
Empate entre los equipos
    3    6
```

Ejemplo. Lea un cuadro de asignación de los paralelos de cada materia, a profesores. Suponer que cada profesor puede tener solo un paralelo de cada materia. Determine:
a) El total de paralelos asignados a cada profesor
b) Lista para cada materia de profesores y paralelos asignadoss

En el cuadro, las filas representarán profesores y las columnas representarán las materias. En el cuadro están los paralelos asignados.

```
a=input('Ingrese matriz de asignación ');
[n,m]=size(a);
% Cantidad de paralelos por profesor
disp('Prof. No. Paralelos')
for  i=1:n                    %asignación por profesor
   c=0;
   for j=1:m
      if a(i,j)>0
         c=c+1;
      end
   end
   disp([i, c]);
end

%Para cada materia, mostrar el profesor y el paralelo asignado
disp('Mat. Profesor Paralelo')
for j=1:m                     %asignación por materia
   for i=1:n
      if a(i,j)>0
         disp([j, i, a(i,j)]);
      end
   end
end
```

```
>> a=[0 2 1 3; 1 1 0 0; 3 3 0 1; 2 0 2 2]
a =
    0    2    1    3
    1    1    0    0
    3    3    0    1
    2    0    2    2
>> cuadro
Ingrese matriz de asignación  a
Prof. No. Paralelos
    1    3
    2    2
    3    3
    4    3
Mat. Profesor Paralelo
    1    2    1
    1    3    3
    1    4    2
    2    1    2
    2    2    1
    2    3    3
    3    1    1
    3    4    2
    4    1    3
    4    3    1
    4    4    2
```

Ejemplo. Generar una tabla ordenada de 5 filas y tres columnas con números aleatorios entre 1 y 25.

```
v=[ ];
while length(v)<15
   x=fix(rand*25)+1;
   if ismember(x,v)==0
      v=[v,x];
   end
end
v=sort(v);
k=1;
t=[ ];
for i=1:5
   for j=1:3
      t(i,j)=v(k);
      k=k+1;
   end
end
disp(t)
```

```
>> tabla
    5    6    8
    9   10   12
   13   14   15
   16   17   20
   21   22   24
```

Ejemplo. Generar el triángulo de Pascal de n filas

```
1
1     1
1     2     1
1     3     3     1
1     4     6     4     1
1     5     10    10    5     1
etc.
```

```
n=input('cuantas filas ');
for i=1:n
   a(i,1)=1;
   a(i,i)=1;
end
for i=3:n
   for j=2:i-1
      a(i,j)=a(i-1,j-1)+a(i-1,j);
   end
end
disp(a)
```

La asignacion de la primera columna y de la diagonal con 1's puede hacerse tambien directamente con las instrucciones:

```
a=eye(n);
a(:,1)=1;
```

Ejemplo. Definir y procesar una matriz con dígitos binarios

```
n=input('tamaño de la matriz ');
a=fix(rand(n,n)*2);
% filas con número par de 1's
for i=1:n
   s=sum(a(i,:));
   if mod(s,2)==0
      disp(i);
   end
end

% fila con más 1's
t=[ ];
for i=1:n
   s=sum(a(i,:));
   t=[t, s];
end
[m,p]=max(t);
disp('fila con más unos');
disp(p);

%convertir a decimal
disp('números en decimal');
for i=1:n
   x=0;
   for j=1:n
      x=x+a(i,j)*2^(n-j);
   end
   disp(x);
end
```

8 Programas que interactúan con un menú

Estructura básica para interactuar con un menú

1) El programa muestra las opciones disponibles para el usuario
2) El usuario elige una opción
3) El programa realiza la acción solicitada

Si la interacción está en un ciclo, el programa debe incluir una opción para salir. Para programar la realización de las acciones, es adecuado el uso de la instrucción **switch**.

Ejemplo. Conversión de Temperatura entre ºF y ºC
La siguiente fórmula permite convertir un valor de temperatura entre grados **farenheit** y grados **celcius**:

$$c = \frac{5}{9}(f - 32)$$

Escriba un programa con un **menú** para realizar la conversión en ambos casos:

```
x=0;
while x~=3
   disp('1) Convertir F a C');
   disp('2) Convertir C a F');
   disp('3) Salir');
   x=input('elija una opción ');
   switch x
      case 1
         f=input('ingrese grados F ');
         c=5/9*(f-32);
         disp(c);
      case 2
         c=input('ingrese grados C ');
         f=9/5*c+32;
         disp(f);
   end
end
```

Ejemplo. Manejo de un conjunto. Escriba un programa para simular algunas operaciones de manejo de un conjunto mediante un menú que dentro de un ciclo ofrezca las siguientes opciones:

1) Agregar un elemento al conjunto
2) Eliminar un elemento del conjunto
3) Determinar si un elemento pertenece al conjunto
4) Salir

Use un vector para almacenar el conjunto. Inícielo como un vector nulo.
En cada opción el programa debe pedir un elemento al usuario y realizar la operación solicitada.

```
v=[];
x=0;
while x~=4
   disp('1) Agregar elemento');
   disp('2) Eliminar elemento');
   disp('3) Pertenecia de un elemento');
   disp('4) Salir');
   x=input('Elija una opcion ');
   switch x
      case 1,
         e=input('ingrese elemento ');
         if ismember(e,v)==0
            v=[v, e];
         end
      case 2,
         e=input('ingrese elemento ');
         [z,p]=ismember(e,v);
         if z == 1
            v(p)=[];
         end
      case 3;
         e=input('ingrese elemento ');
         z=ismember(e,v);
         disp(z);
   end
end
```

Ejemplo. Un programa con un **menú** y una **repetición** para mostrar el valor de la compra de una pizza. Elegir el tamaño de la pizza y luego ingresar el número de ingredientes adicionales que se desean. Calcular el valor a pagar sabiendo que la pizza pequeña cuesta $**6,** la grande cuesta $**8** y la familiar **$12.** Cada ingrediente adicional cuesta $**1.20.** Incluya el IVA.

t: tamaño de la pizza
n: número de ingredientes
p: valor a pagar

```
op=0;
while op ~= 4
        disp('1) Pizza pequeña ');
        disp('2) Pizza grande ');
        disp('3) Pizza familiar ');
        disp('4) Salir ');
        op=input('Elija una opción ');
        switch op
           case 1,
                  n=input('Número de ingredientes ');
                  p= 1.12*(6+n*1.2);
                  disp(p);
           case 2,
                  n=input('Número de ingredientes ');
                  p= 1.12*(8+n*1.2);
                  disp(p);
           case 3,
                  n=input('Número de ingredientes ');
                  p= 1.12*(12+n*1.2);
                  disp(p);
           otherwise,
                  disp('Error');
        end
end
```

Agregue comandos **clc** y **pause** para mejorar la interacción

Ejemplo: Uso interactivo de la fórmula del interés compuesto con un menú.

$$A = P\left[\frac{(1+x)^n - 1}{x}\right]$$, en donde

A: Valor acumulado, **P**: Valor de cada depósito **mensual**

n: Cantidad de depósitos **mensuales** **x**: Tasa de interés **mensual**

Diseñar un algoritmo que permita interactuar con un **menú** para obtener; **A, P, n**

```
format bank;
t=0;
while t~=4
   disp('1) Valor acumulado');
   disp('2) Depósito mensual');
   disp('3) Número de depósitos');
   disp('4) Salir');
   t=input('Elija una opción ');
   switch t
      case 1,
         p=input('Valor del depósito mensual ');
         n=input('Número de depósitos mensuales ');
         x=input('Interés anual en porcentaje ');
         m=0.01*x/12;
         a=p*((1+m)^n-1)/m;
         disp('Valor acumulado ');
         disp(a)
      case 2,
         a=input('Valor acumulado ');
         n=input('Número de depósitos mensuales ');
         x=input('Interés anual en porcentaje ');
         m=0.01*x/12;
         p=a*m/((1+m)^n-1);
         disp('Cuota mensual ');
         disp(p)
      case 3,
         a=input('Valor acumulado ');
         p=input('Valor del depósito mensual ');
         x=input('Interés anual en % ');
         m=0.01*x/12;
         n=log(a*m/p+1)/log(1+m);
         disp('Numero de depósitos ');
         disp(n)
   end
end
```

Interacción con MATLAB.

1) Valor acumulado
2) Depósito mensual
3) Número de depósitos
Elija una opción 1

Valor del depósito mensual 120
Número de depósitos mensuales 60
Interés anual en % 4
Valor acumulado
7955.88

Ejercicio: Programa con un menú para usar la fórmula de la suma de una sucesión aritmética

La suma de los términos de una sucesión aritmética está dada por

$s = \frac{n}{2}(a+u)$, en donde

s: Suma de los términos,
n: Cantidad de términos
a: Primer término de la sucesión,
u: Último término de la sucesión

El menú permitirá calcular: **s, n, a, u** dados los otros 3 datos.

La interacción debe estar en un ciclo con una opción para salir:

Menú
1) Suma de términos
2) Cantidad de términos
3) Primer término
4) Ultimo término

```
x=0;
while x~=5
   disp('1) Suma de términos');
   disp('2) Cantidad de términos');
   disp('3) Primer término');
   disp('4) Último término');
   disp('5) Salir');
   x=input('Elija una opción ');
   switch x
      case 1
         n=input('Cuantos términos ');
         a=input('Primer término ');
         u=input('Último término ');
         s=n/2*(a+u);
         disp(s);
      case 2

      case 3

      case 4

   end
end
```

Completar las opciones

Ejemplo. Escriba un programa para control de venta de las sillas de un evento con las siguientes opciones:

1) Verificar silla disponible
2) Venta de silla
3) Devolución de silla

```
n=input('cuantas filas ');
m=input('cuantas columnas ');
s=zeros(n,m);
x=0;
while x~=4
   disp('1) verificar silla');
   disp('2) venta');
   disp('3) devolución');
   disp('4) salir');
   x=input('elija una opción ');
   switch x
      case 1
         f=input('cual fila ');
         c=input('cual columna ');
         if s(f,c)==0
            disp('silla disponible');
         else
            disp('silla ocupada');
         end
      case 2
         f=input('cual fila ');
         c=input('cual columna ');
         if s(f,c)==1
            disp('silla ocupada');
         else
            s(f,c)=1;
         end
      case 3
         f=input('cual fila ');
         c=input('cual columna ');
         s(f,c)=0;
   end
end
```

Ejemplo. Programa para control de los casilleros de un club. Los casilleros están organizados en filas y columnas.

Si el casillero está libre contiene cero. Si está asignado, contiene el código del usuario.

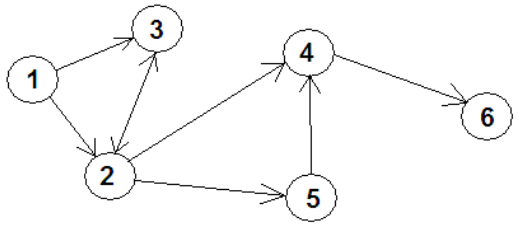
```
n=input('cuantas filas ');
m=input('cuantas columnas ');
c=zeros(n,m);
x=0;
while x~=4
  disp('1) Asignar');
  disp('2) Devolver');
  disp('3) Consultar');
  disp('4) Salir');
  x=input('Elija una opción ');
  switch x
    case 1
      i=input('cual fila ');
      j=input('cual columna ');
      if c(i,j)~=0
        disp('casillero ocupado');
      else
        s=input('ingrese el código del socio ');
        c(i,j)=s;
      end
    case 2




    case 3




  end
end
```

Completar las opciones

8.1 Ejercicios con matrices y uso de menú

1.- El área de un patio está distribuida en celdas ordenadas en filas y columnas. En cada celda están almacenados paquetes del mismo tipo.
Escriba un programa que lea una matriz cuyo contenido representa la cantidad de paquetes ubicados en cada celda
a) Encuentre el valor promedio de la cantidad de paquetes existentes en todas las celdas.
b) Encuentre cual celda contiene más paquetes.
c) Muestre las coordenadas y la cantidad de paquetes que contiene una celda cuya posición: número de fila y número de columna, son valores elegidos al azar en el programa..

2.- Un grafo consta de vértices que pueden representarse mediante círculos y arcos que los conectan. Esta conectividad puede describirse mediante una matriz en la que el valor **1** indica que existe un arco en esa dirección, mientras que el valor **0** indica que no existe el arco con esa dirección, como se muestra en el ejemplo



Matriz de conectividad

	1	**2**	**3**	**4**	**5**	**6**
1	1	1	1	0	0	0
2	0	1	1	1	1	0
3	0	1	1	0	0	0
4	0	0	0	1	0	1
5	0	0	0	1	1	0
6	0	0	0	1	0	1

Escriba un programa para almacenar 0 o 1 aleatoriamente en una matriz nxn, siendo n un dato que debe leerse inicialmente. Dentro del programa llene la diagonal con 1's para indicar que cada nodo está conectado consigo mismo. El programa examinar las filas para determinar
a) Cual nodo no tiene conecciones con otros nodos (no tiene arcos)
b) Cual es el nodo que tiene más conecciones con otros nodos

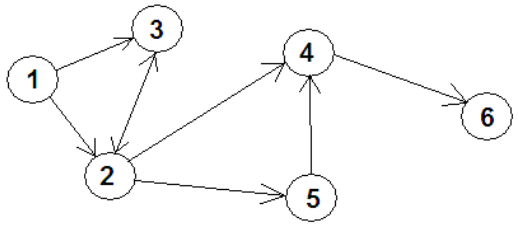
3.- Lea una matriz nxn, siendo n un dato inicial. Suponga que cada celda contiene un dato (peso en kg.). Determine cuales son las celdas interiores, es decir que no debe considerar las celdas en los bordes, en las cuales el valor del peso es mayor que el promedio de las cuatro celdas ubicadas a sus cuatro lados inmediatos.

4.- Diseñe un programa para administrar el uso de los casilleros de una institución. Los casilleros están organizados en n filas y m columnas. Inicialmente los casilleros contienen el valor cero, lo cual significa que están vacíos. El programa debe usar un menú con las siguientes opciones:

1) Consultar casillero
2) Asignar casillero
3) Devolver casillero
4) Buscar usuario
5) Salir

En la opción 1, verificar si el casillero contiene cero, mostrar mensaje DISPONIBLE u OCUPADO
En la opción 2, almacenar en el casillero el código del usuario asignado
En la opción 3, almacenar cero en el casillero que es devuelto
En la opción 4, ingresar el código de algún usuario. Buscar la ubicación del casillero asignado.

5.- Un grafo consta de vértices que pueden representarse mediante círculos y arcos que los conectan. Esta conectividad puede describirse mediante una **matriz** en la que el valor **1** indica que existe un arco en esa dirección, mientras que el valor **0** indica que no existe el arco con esa dirección, como se muestra en el ejemplo



Matriz de conectividad

	1	**2**	**3**	**4**	**5**	**6**
1	1	1	1	0	0	0
2	0	1	1	1	1	0
3	0	1	1	0	0	0
4	0	0	0	1	0	1
5	0	0	0	1	1	0
6	0	0	0	1	0	1

Escriba un programa para manejar interactivamente la conectividad de un grafo mediante un menú con las siguientes opciones de un menú:

1) Agregar arco
2) Eliminar arco
3) Consultar arco
4) Vértices libres
5) Salir

Al inicio debe pedir el número de vértices n. Llenar con **1** la diagonal y **0** en el resto de la matriz nxn

En la opción 1) debe pedir los vértices inicial y final, y colocar **1** en la celda de la matriz ubicada en la fila y columna respectivas.
En la opción 2) debe pedir los vértices inicial y final, y colocar **0** en la celda de la matriz ubicada en la fila y columna respectivas.
En la opción 3) debe pedir los vértices inicial y final, y mostrar un mensaje dependiendo del contenido de la celda de la matriz ubicada en la fila y columna respectivas
En la opción 4) busque cada fila (vértice inicial) que tiene todas las columnas con ceros (vértices finales), excepto la diagonal

6.- Desarrolle una aplicación basada en vectores para instrumentar las operaciones básicas entre dos conjuntos A, B mediante las siguientes opciones de un menú:

1) Agregar elemento
2) Eliminar elemento
3) Pertenencia
4) Consultar
5) Unión
6) Intersección
7) Diferencia
8) Salir

En la opción 1) preguntar cual conjunto: A o B y luego pedir el elemento para agregar
En la opción 2) preguntar cual conjunto: A o B y luego pedir el elemento para eliminar
En la opción 3) pedir el elemento y determinar en cual conjunto se encuentra (pueden ser ambos)
En la opción 4) preguntar cual conjunto: A o B y luego mostrar sus elementos

9 Funciones en MATLAB

En general una función en los lenguajes de programación es un conjunto de instrucciones que se escriben separadamente del programa y que realizan alguna tarea especificada. Los usuarios pueden definir funciones y agregarlas a las funciones propias de MATLAB.

El mecanismo usual para trasmitir datos a las funciones es mediante una lista de variables que se denominan parámetros. Sin embargo, a diferencia de los programas, las variables que se usan dentro de una función, no están disponibles fuera de ella, a menos que se use una declaración explícita y que se verá mas adelante.

Declaración de una función en MATLAB

```
function  variable = nombre (parámetros)
    instrucciones
```

variable contendrá el resultado que entrega la función
parámetros son variables que reciben los datos que entran a la función.
nombre identificación de la función
instrucciones se incluyen en la función según la tarea especificada

El nombre asignado a una función debe coincidir con el nombre usado en la declaración de la función.

Las funciones se escriben en la ventana de edición de MATLAB y se las almacena en alguna carpeta. En la ventana de comandos debe especificarse la ruta a esta carpeta.

El uso de una función es similar al uso de las funciones comunes de MATLAB. El nombre debe coincidir con el nombre asignado, aunque los parámetros pueden tener nombres diferentes, pero su uso debe ser coherente.

El concepto de función es fundamental en matemática y MATLAB proporciona un dispositivo natural para su instrumentación

Ejemplo: Instrumentar en MATLAB la siguiente función de variable real:

$$x \rightarrow \boxed{y = f(x) = 2x^2 + 1} \rightarrow y$$

Instrumentación en MATLAB

```
function y=f(x)
y=2*x^2 + 1;
```

```
>> y=f(2)
    y = 9
```

Ejemplo. Escriba una función para elegir el mayor entre dos números

Abra un documento nuevo en la ventana de edición y escriba:

```
function m = mayor(a, b)
if a>b
  m = a;
else
  m = b;
end
```

m es la variable que entrega el resultado
mayor es el nombre de la función
a, b son los parámetros que ingresan los datos a la función

Almacene esta función con el nombre **mayor**
Suponer que quiere escoger el mayor entre e^{π} y π^{e}.
Escriba en la ventana de comandos:

```
>> a = exp(pi);
>> b = pi^exp(1);
>> m = mayor(a, b)
        23.1407                (respuesta que muestra MATLAB)
```

Los nombres de las variables pueden ser diferentes:

```
>> x = exp(pi);
>> y = pi^exp(1);
>> t = mayor(x, y)
        23.1407                (respuesta que muestra MATLAB)
```

Ejemplo. Escriba una función que reciba un número y determine la cantidad de divisores enteros exactos que tiene el número.

```
function  c = ndiv( n )
c = 0;
for d = 1: n
  if mod(n, d) == 0
    c = c + 1;
  end
end
```

Guarde la función en el disco con el nombre **ndiv**
Pruebe la función directamente desde la ventana de comandos

```
>> n = 25;
>> c = ndiv(n)
        3                      (resultado que muestra MATLAB)
>> c = ndiv(43)
        2                      (resultado que muestra MATLAB)
```

Ejemplo. Escriba una función que reciba un número y determine si es un número primo. El resultado que entrega la función será 1 o 0 según corresponda;

```
function  p = primo( n )
c = 0;
for d = 1: n
  if mod(n, d) == 0
    c = c + 1;
  end
end
if c > 2
  p = 0;
else
  p = 1;
end
```

Guarde la función en el disco con el nombre **primo**
Pruebe la función directamente desde la ventana de comandos

```
>> n = 25;
>> p = primo(n)
        0                    (resultado que muestra MATLAB)
>> p = primo(43)
        1                    (resultado que muestra MATLAB)
```

Ejemplo. Escriba en una nueva ventana de edición un programa que use la función **primo** para encontrar todos los números primos menores a 20:

```
for n = 1: 20
  if primo(n) == 1
    disp(n);
  end
end
```

Salida

```
        1                    (resultados mostrados por MATLAB)
        2
        3
        5
        7
        11
        13
        17
        19
```

Ejemplo. Escriba un programa que use la función **primo** para encontrar todas las parejas de números primos cuya suma sea igual a un número par dado:

```
n=input('ingrese un numero par');
for a=1:n
   for b=a:n
      if primo(a)==1 & primo(b)==1 & a+b == p
         disp([a,b]);
      end
   end
end
```

Prueba del programa

```
ingrese un numero 50
    3    47
    7    43
   13    37
   19    31
```

Si es necesario verificar que el dato es un número par se puede incluir una validación:

```
res=1;
while res~=0
    n=input('ingrese un numero par');
    res=mod(n,2);
end
for a=1:n
   for b=a:n
      if primo(a)==1 & primo(b)==1 & a+b == p
         disp([a,b]);
      end
   end
end
```

Una función puede entregar más de un resultado
Las variables que entregan los resultados deben definirse entre corchetes.

Ejemplo. Escriba una función que entregue el área y el volumen de un cilindro dados su radio (r) y su altura (h)

```
function [a, v] = cilindro(r, h)
a = 2*pi*r*h + 2*pi*r^2;
v = pi*r^2*h;
```

Escriba y almacene la función con el nombre cilindro.

Use la función para calcular el área y el volumen de una lata de cilíndrica que tiene un diámetro de 10cm y una altura de 12cm

Escriba en la ventana de comandos:
>> r = 5;
>> h = 12;
>> [a, v] = cilindro(r,h) Se muestran ambos resultados
>> a = cilindro(r,h) Se muestra el primer resultado

9.1 Variables locales y variables globales

Las variables definidas dentro de una función son locales, es decir que a diferencia de los programas, no son visibles fuera de la función

Ejemplo. Escriba la función:

```
function  x=fn(a, b)
c = a + b;
x = 2*c;
```

Almacenar con el nombre **fn** y usar desde la ventana de comandos:

```
>> a = 3;
>> b = 5;
>> t = fn(a, b)
        t = 16          (resultado que muestra MATLAB)
>> c                    (intentamos conocer el valor de c en la función)
```

??? Undefined function or variable 'c'.

mensaje de error de MATLAB que indica que **c** no está definida

Las variables en los programas son visibles (disponibles) fuera del programa.

Ejemplo. Un programa con acción similar a la función anterior:

```
a = input('ingrese dato ');
b = input('ingrese dato ');
c = a + b;
x = 2*c;
disp(x);
```

Almacene con el nombre prueba y active el programa:

```
>> prueba
        ingrese dato  3        (interacción para ingreso de datos)
        ingrese dato  5
           16                  (resultado que muestra MATLAB)
>> c
        c = 8                  (la variable c puede ser utilizada)
```

Es posible hacer que las variables de una función sean visibles fuera de su ámbito mediante una declaración especial

El comando **global** permite tener acceso a variables de las funciones

Ejemplo. Modifique la función **fn** para que la variable **c** sea visible:

```
function  x=fn(a, b)
global c;
c = a + b;
x = 2*c;
```

Almacene con el nombre fn y use la función:

```
>> global c;
>> a = 3;
>> b = 5;
>> t = fn(a, b)
        t = 16          (resultado que muestra MATLAB)
>> c                    (intentamos conocer el valor c de la función)
        c=8             (la variable c está disponible ahora)
```

Una función puede no necesitar parámetros

Ejemplo. Escriba una función que lea y valide un entero entre 1 y 5

```
function n=entero
x=0;
while x==0
  n=input('ingrese un entero entre 1 y 5 ');
  if n>0 & n<6
    x=1;
  end
end
```

Una función puede no entregar resultados ni usar parámetros

Ejemplo. Escriba una función que muestre un menú en la pantalla

```
function menú
clc;
disp('1) ingresar');
disp('2) borrar');
disp('3) salir');
```

para usar esta función escriba

>> menu

Una función puede recibir como parámetros vectores o matrices.

Ejemplo. Escriba una función que reciba un vector y entregue el promedio del valor de sus elementos.

```
function p=prom(x)
n=length(x);
s=0;
for i=1:n
  s=s+x(i);
end
p=s/n;
```

Esta función es equivalente a la función **mean** de MATLAB

Para usar la función creada debe definir el vector antes de llamarla
La longitud del vector se determina con la función **length** de MATLAB

```
>> x=[2 7 3 5 4 7 6];
>> t=prom(x)
```

t = 4.8571 (es el resultado que muestra MATLAB)

Ejemplo. Escriba una función que reciba un vector y entregue la suma de los valores pares únicamente.

```
function s=sumapares(x)
n=length(x);
s=0;
for i=1:n
  if mod(x(i), 2) == 0
    s=s+x(i);
  end
end
```

```
>> x=[2 7 3 5 4 7 6];
>> t=sumapares(x)
```

t = 12 (es el resultado que muestra MATLAB)

Una función puede recibir y entregar vectores o matrices

Ejemplo. Escriba una función que reciba un vector y entregue otro vector conteniendo los elementos cuyo valor es un número par.

```
function y=pares(x)
n=length(x);
y=[ ];
for i=1:n
  if mod(x(i), 2) == 0
     y = [y, x(i)];
   end
end
```

```
>> x=[2 7 3 5 4 7 6];
>> t=pares(x)
    t =
        2  4  6
```

(es el resultado que muestra MATLAB)

Ejemplo. Escriba una función que entregue un vector de longitud n conteniendo números aleatorios enteros con valor entre 1 y 6:

```
function d=dados(n)
for i=1:n
   d(i)=fix(rand*6+1);
end
```

Para usar esta función debe enviar un valor para el parámetro n:

```
>> t=dados(5)
        t = 6   3   4   3   2
```

(es un vector resultante de MATLAB)

Una solución alternativa para el ejemplo anterior:

```
function d=dados(n)
d=[ ];
while length(d) < n
   t =  fix(rand*6+1);
   d = [d, t]
end
```

Ejemplo. Escriba una función para emular la función **max** de MATLAB

```
function [m,p]=mayor(x)
n=length(x);
m=x(1);
p=1;
for i=2:n
   if x(i)>m
      m=x(i);
      p=i;
   end
end
```

```
>> x=[6 7 9 2 8 3];
>> m=mayor(x)
m =
    9
>> [m, p]=mayor(x)
m =
    9
p =
    3
```

Ejemplo. Escriba una función para emular la función **ismember** de MATLAB

```
function [r,p]=pertenece(e,v)
n=length(v);
r=0;
p=0;
for i=1:n
   if e==v(i)
      r=1;
      p=i;
   end
end
```

```
>> v=[6 7 9 2 8 7 3];
>> r=pertenece(7,v)
r =
    1
>> [r,p]=pertenece(7,v)
r =
    1
p =
    6
```

Ejemplo. Una función para eliminar elementos repetidos

```
function y=norep(x)
n=length(x);
y=[ ];
for i=1:n
   if pertenece(x(i),y)==0
      y=[y, x(i)];
   end
end
```

Ejemplo. Escriba una función que reciba dos conjuntos (vectores) A, B y entregue un tercer conjunto (vector) que contenga los elementos que están en ambos conjuntos (vectores):

```
function c=interseccion(a, b)
n=length(a);
m=length(b);
k=1;
for i=1:n
   for j=1:m
      if a(i) == b(j)
    c(k) = a(i);
         k = k + 1;
      end
   end
end
c=norep(c);              %Elimina elementos repetidos
```

Para usar esta función debe definir los vectores que entran. Recuerde que pueden tener nombres diferentes a los que usa la función:

```
>> a=[2 7 5 4 3 8];
>> b=[7 1 3 9 0];
>> c=interseccion(a, b)
  c =    7    3  (Es el vector resultante que entrega MATLAB)
```

Una forma alterna para escribir la función anterior

```
function c=interseccion(a, b)
c=[ ];
for i=1: length(a)
   for j=1: length(b)
      if a(i) == b(j)
            c = [c , a(i)];
      end
   end
end
c=norep(c);              %Elimina elementos repetidos
```

Una forma más eficiente para escribir la función anterior

```
function c=interseccion(a, b)
n=length(a);
c=[ ];
for i=1:n
   if pertenece(a(i), b)
       c=[c, a(i)];
   end
end
c=norep(c);              %Elimina elementos repetidos
```

Unión de conjuntos

```
function c=union(a,b)
m=length(b);
c=a;
for i=1:m
   if ~pertenece(b(i),c)
      c=[c,b(i)];
   end
end
c=norep(c);              %Elimina elementos repetidos
```

Versión más eficiente

```
function c=union(a,b)
c=[a, b];
c=filtro(c);
```

Diferencia de conjuntos

```
function c=diferencia(a, b)
n=length(a);
c=[ ];
for i=1:n
   if ~pertenece(a(i), b)
      c=[c, a(i)];
   end
end
c=norep(c);              %Elimina elementos repetidos
```

Ejemplo. Defina una función para elegir una muestra aleatoria de tamaño **n** de una población de tamaño **m.** Pueden haber elementos repetidos.

```
function t=muestra(m, n)
t=[ ];
for i=1:n
   x=fix(rand*m)+1;
   t=[t, x];
end
```

Ejemplo. Defina una función para elegir una muestra aleatoria de tamaño **n** de una población de tamaño **m**. Los elementos **no deben estar repetidos** en la muestra

```
function t=muestra(m,n)
t=[ ];
while length(x)<n
   x=fix(rand*m)+1;
   if ~pertenece(x, t)
      t=[t, x];
   end
end
```

Uso de la función muestra

```
>> x=muestra(10,4)
x =
   5    10     9     6
```

Ejemplo. Escriba una función para obtener una tabla de 15 elementos (diferentes) con números aleatorios entre 1 y 25.

```
function t=tabla
v=[ ];
while length(v)<15
   x=fix(rand*25)+1;
   if pertenece(x,v)==0
      v=[v, x];
   end
end
r=sort(v);
k=1;
for i=1:5
   for j=1:3
      t(i,j)=r(k);
      k=k+1;
   end
end
```

```
>> t=tabla
t =
    1    2    3
    4    7   11
   13   14   15
   16   17   18
   19   20   22
```

Si la salida de una función es antes del final, se debe usar **return**

Ejemplo. Escriba una función para determinar si los elementos de un vector están en orden creciente:

Si el resultado es 1, los elementos están en orden creciente, caso contrario, el resultado es 0

```
function t=orden(x)
n=length(x);
for i=1:n-1
   if x(i) > x(i+1)
         t=0;
         return;
   end
end
t=1;
```

9.2 Programas que llaman a funciones

Ejemplo. Una funcion para ordenar un vector en forma creciente

```
function x=ordenar(x)
n=length(x);
for i=n:-1:2
   [t,p]=max(x(1:i));
   [x(i),x(p)]=cambiar(x(i),x(p));
end

function [a,b]=cambiar(a,b)
t=a;
a=b;
b=t;
```

Ejemplo. Una función para eliminar espacios intermedios de una frase:

f: contiene una frase que entra a la función
x contiene la frase sin espacios intermedios

```
function x=compactar(f)
n=length(f);
x='';
for i=1:n
   if f(i) ~= ' '
      x = strcat(x, f(i));
   end
end
```

Una forma alterna de escribir la función anterior

```
function x=compactar(f)
n=length(f);
x='';
for i=1:n
   if f(i) ~= ' '
      x = [x, f(i)];
   end
end
```

Ejemplo. Un programa que lee una frase, usa la función **compactar** para eliminar los espacios intermedios, y luego muestra un mensaje en caso de que la frase sea simétrica: sus caracteres opuestos son iguales

```
f=input('ingrese una frase ');
f=compactar(f);
n=length(f);
sim=1;
for i=1:n/2
   if f(i) ~= f(n-i+1)
      sim=0;
   end
end
if sim == 1
   disp('la frase es simetrica');
else
   disp('la frase no es simetrica');
end
```

Probamos este programa suponiendo que lo hemos almacenado con el nombre **prueba**:

```
>> prueba
ingrese una frase 'anita lava la tina';          (dato que ingresa)

        la frase es simetrica                    (resultado de MATLAB)
```

MATLAB tiene la función **error** para desplegar mensajes de error y terminar la ejecución:

Ej.

```
if d<0
   error('valor incorrecto');
end
```

Ejemplo. Para cada número natural existe al menos otro número natural tal que el resultado de multiplicar estos dos números solamente tiene ceros y unos.
Ejemplo: El número **6** multiplicado por **185**, el resultado es **1110**
El número **38** multiplicado por **2895**, el resultado es **110010**
El número **253** multiplicado por **43917**, el resultado es **11111001**
Etc.
Escriba una función **n = factor(t)** que reciba un número natural **t** y entregue otro número natural **n**, tal que el producto **t** por **n** solamente tenga ceros y unos, como en los ejemplos anteriores.

```
function n=factor(t)
p=1;
n=1;
while p>0
   p=t*n;
   while p>0
      d=mod(p,10);
      if d>1
         n=n+1;
         break;
      else
         p=fix(p/10);
      end
   end
end
```

```
Ejm.
>> n=factor(237)
n =
     426203
>> n*237
ans =
  101010111
```

Este algoritmo es una búsqueda exhaustiva simple

9.3 Funciones recursivas

Una función puede llamarse a si misma. Estas funciones se denominan recursivas.

Ejemplo. La siguiente definición suma los cubos de los primeros n números naturales:

$$sc(n) = \begin{cases} sc(n-1) + n^3, & n > 1 \\ 1, & n = 1 \end{cases}$$

Se puede instrumentar directamente en MATLAB:

```
function r=sc(n)
if n>1
    r=sc(n-1)+n^3;
else
    r=1;
end
```

Use la función:

```
>> s=sc(5)
   s =
        225
```

Ejemplo. Use la siguiente definición recursiva para calcular el máximo común divisor entre dos números enteros. Escriba una función con esta definición

$$mcd(a,b) = \begin{cases} mcd(a-b,b), & a > b \\ mcd(a,b-a), & b > a \\ a, & a = b \end{cases}$$

```
function c=mcd(a, b)
if a>b
    c=mcd(a-b, b);
else
      if b>a
            c=mcd(a, b-a);
      else
            c=a;
      end
end
```

Use la función:

```
>> x=mcd(36, 48)
```

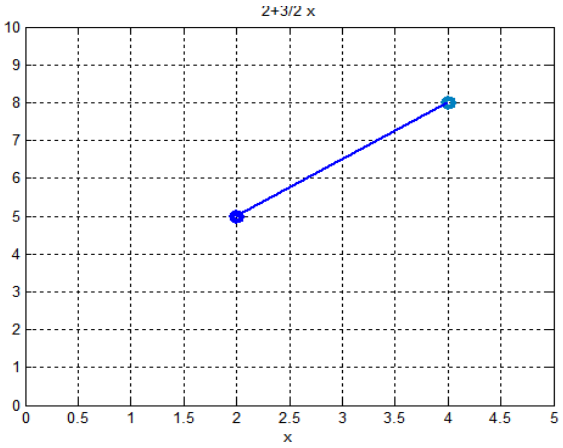
Funciones que entregan resultados analíticos

Ejemplo. Escriba y almacene una función que reciba dos puntos y entregue la ecuación de la recta que incluye a estos dos puntos:

```
function y=recta(x1, y1, x2, y2)
syms x;
m=(y2 - y1)/(x2 - x1);
y=y1 + m*(x - x1);
```

Uso de la función desde la línea de comandos y su gráfico

```
>> x1=2;
>> y1=5;
>> x2=4;
>> y2=8;
>> y=recta(x1, y1, x2, y2)
        y =
             2+3/2*x
>> plot(x1,y1,'o',x2,y2,'o')
>> hold on
>> ezplot(y, [2,4])
>> grid on
>> axis([0,5,0,10])
>>
```



9.4 Ejercicios con funciones

1. Escriba una función **conteo(n)** que entregue la **cantidad** de divisores enteros positivos que tiene un número entero dado n. Escriba un programa de prueba que use la función escrita para encontrar cual número entre 1 y 100 tiene más divisores enteros.

2. Escriba una función **primo(n)** para determinar si un número **n** dado es primo. Escriba un programa de prueba que mediante la función escrita, encuentre los números primos existentes entre 1 y **n**, siendo **n** un dato.

3. Escriba una función **primo(n)** para determinar si un número **n** dado es primo. Escriba un programa de prueba que use la función primo y encuentre dos números enteros aleatorios menores que 100 tales que su suma sea también un número primo.

4. Escriba una función **divisores(n)** que entregue un vector conteniendo todos los divisores enteros positivos que tiene un número entero dado n. Escriba un programa de prueba que use la función escrita para encontrar para cada número entero del 20 al 30, sus divisores enteros

5. Escriba una función **mayor(x)** que reciba un vector **x** y devuelva el mayor valor. Escriba un programa de prueba que genere y almacene en un vector **n** números aleatorios entre 1 y 100. Use la función escrita y encuentre y muestre el mayor valor generado.

6. Escriba una función **perfecto(n)** que determine si un número entero dado n es un número perfecto. Un número perfecto debe ser igual a la suma de todos sus divisores enteros menores que el valor del número.
   Ejemplo: 28 = 1 + 2 + 4 + 7 + 14
   Escriba un programa de prueba que use la función escrita y encuentre los números perfectos entre 1 y 1000

7. Escriba una función **suma(n)** que entregue la suma de las cifras de un número dado **n**. Con esta función escriba un programa que genere 10 números aleatorios entre 1 y 100 y encuentre cual de ellos tiene la mayor suma de sus cifras.

8. Escriba una función **cuad(n)** que determine si el cuadrado de un número natural n dado, es igual a la suma de los primeros n números impares.
   Ej. $6^2 = 1+3+5+7+9+11$
   Escriba un programa de prueba que ingrese un dato desde el teclado, use la función y muestre el resultado en la pantalla.

9. Escriba una función **secuencia(n)** que entregue el n-ésimo término de la siguiente secuencia, en la cual cada término, a partir del tercero se obtiene sumando los dos anteriores: 1, 1, 2, 3, 5, 8, 13, 21, ... Escriba un programa de prueba que ingrese un dato desde el teclado use la función y muestre el resultado en la pantalla.

10. Escriba una función **conteo(x)** que reciba una cadena de caracteres x, y determine la cantidad de palabras que contiene. Suponga que las palabras están separadas por un espacio. Escriba un programa de prueba que ingrese un dato desde el teclado, use la función y muestre el resultado en la pantalla.

11. Escriba una función **sim(x)** que reciba un entero y determine si es simétrico, es decir si los dígitos opuestos alrededor del centro son iguales. Escriba un programa de prueba que genere números aleatorios entre 1 y 10000 hasta obtener un número que sea simétrico

```
function s=sim(x)
t=x;
y=0;
while t>0
```

```
    d=mod(t,10);
    y=y*10+d;
    t=fix(t/10);
end
if x == y
    s=1;
else
    s=0;
end
```

12. Escriba una función **alfin(n)** que entregue como resultado la cantidad de veces que debe lanzarse un dado hasta que salga un número n dado como parámetro. Escriba un programa de prueba que ingrese un dato desde el teclado, use la función y muestre el resultado en la pantalla.

13. Escriba una función **conteo(x)** que determine la cantidad de términos que deben sumarse de la serie: 1*2*3 + 2*3*4 + 3*4*5 + 4*5*6+..... hasta que la suma exceda a un valor x dado. Escriba un programa de prueba que genere un número aleatorio para x entre 1 y 1000, use la función y muestre el resultado en la pantalla.

14. Escriba una función recursiva **suma(n)** que retorne la suma de los cubos de los primeros n números naturales, Escriba un programa de prueba en el cual el dato n entregado a la función es un número aleatorio entre 1 y 10. Muestre el resultado.

15. Escriba una función **secuencia(n)** que entregue el n-ésimo término de la siguiente secuencia, en la cual cada término, a partir del cuarto se obtiene sumando los tres anteriores: 1, 1, 1, 3, 5, 9, 17, 31, 57, ... Escriba un programa de prueba que ingrese un dato desde el teclado, use la función y muestre el resultado en la pantalla.

16. Escriba una función **fact(n)** que reciba un numero entero n y devuelva su factorial. Escriba un programa de prueba que genere un número aleatorio entero menor que 8, use la función y muestre la suma de los factoriales de los primeros k números naturales

17. Escriba una función **suma(n)** que reciba un numero entero n y devuelva la suma de sus dígitos. Escriba un programa de prueba que genere números aleatorio entre 1 y 100 hasta que la suma de los dígitos de alguno de ellos sea múltiplo de 7

18. Escriba una función **suma(n)** que reciba un número entero n y devuelva la suma de sus divisores. Escriba un programa de prueba que ingrese un dato desde el teclado, use la función y muestre el resultado en la pantalla

19. Escriba una función **mcd(a, b)** que reciba dos números enteros a y b, y devuelva el máximo común divisor entre ellos. Escriba un programa de prueba que genere dos números aleatorios entre 1 y 100, use la función y muestre el resultado en la pantalla

20. Escriba una función que reciba un vector y entregue la suma de los elementos que son números primos.

21. Escriba una función que reciba un vector y entregue otro vector conteniendo los elementos que son números primos.

10 Desarrollo de aplicaciones en MATLAB

Ejemplo. Desarrollo en varias versiones de una aplicación para manejo de datos simples con un menú con las opciones:

1) Agregar
2) Consultar
3) Eliminar

VERSION 1: Acciones incluidas dentro del programa

Instrumentación inicial: Las acciones son desarrolladas dentro del programa. Se usa función **ismember** de **MATLAB** para determinar cual celda contiene a un dato dado. Los datos son valores simples almacenados en un vector.

x: vector para almacenamiento de datos
ac: opción del usuario para control de acciones

```
%Programa
x=[ ];
ac=0;
while ac ~= 4
   clc;
   disp('1) Agregar');
   disp('2) Consultar');
   disp('3) Eliminar');
   disp('4) Salir');
   ac=input('Elija una acción ');
   switch ac
      case 1,   t=input('Ingrese dato ');
                [e,p]=ismember(t,x);
                if e==0
                  x=[x, t];
                else
                  disp('Ya existe el dato ');
                  pause;
                end
      case 2, t=input('Ingrese dato ');
                [e,p]=ismember(t,x);
                if e==0
                   disp('No está almacenado');
                   pause;
                else
                   disp('Si está almacenado');
                   pause;
                end
      case 3, t=input('Ingrese dato ');
                [e,p]=ismember(t,x);
                if e==0
                   disp('No está almacenado ');
                   pause;
                else
                   x(p)=[ ];
                end
      end
end
```

VERSION 2:

Acciones almacenadas en funciones y variables globales

x: vector para almacenamiento de datos

ac: opción elegida por el usuario para control de acciones

Se usará la función **pertenece** instrumentada antes en lugar de la función **ismember**

```
%Programa
global x
x=[ ];
ac=0;
while ac ~= 4
   menu1;
   ac=input('Elija una acción ');
   switch ac
      case 1, agregar;
      case 2, consultar;
      case 3, eliminar;
   end
end
```

```
function menu1
clc;
disp('1) Agregar');
disp('2) Consultar');
disp('3) Eliminar');
disp('4) Salir');
```

```
function agregar
global x;
t=input('Ingrese dato ');
[e,p]=pertenece(t,x);
if e==0
  x=[x, t];
else
  disp('Ya existe este dato ');
  pause;
end
```

```
function consultar
global x;
t=input('Ingrese dato ');
[e,p]=pertenece(t,x);
if e==0
   disp('No está almacenado');
   pause;
else
   disp('Si está almacenado');
   pause;
end
```

```
function eliminar
global x;
t=input('Ingrese dato ');
[e,p]=pertenece(t,x);
if e==0
   disp('El dato no está almacenado ');
   pause;
else
   x(p)=[ ];
end
```

```
function [r,p]=pertenece(e,v)
n=length(v);
r=0;
p=0;
for i=1:n
   if e==v(i)
      r=1;
      p=i;
   end
end
```

10.1 Una función para dtectar errores en la ejecución

La instrucción **try-catch-end** permite capturar errores en la ejecución para tomar alguna acción en caso de que se produzcan, evitando así que el programa finalice con una condición de error:

```
try
   instrucciones en las que se quiere detectar una condición de error
catch
   Acción que se realizará en caso de producirse el error
end
```

Ejemplo. Prevenir que una función no exista

```
x=input('Dato ');
try
  r=frst(x);
  disp(r);
catch
   disp('Error');
end
```

Ejemplo. Prevenir que un archivo no esté almacenado en el disco

```
try
  load arch1 x;
catch
  x=[ ];
end
```

10.2 Almacenamiento y recuperación de archivos de datos en el disco

Almacenar variables (y su contenido) en un archivo

save archivo variables

Ejemplo. Guardar las variables a y b en un archivo arch1 en el disco

```
>> a=[4 6 8 3 5];
>> b=[5 7 6; 9 8 4; 3 2 9];
>> save arch1 a b
```

Recuperar las variables (y su contenido) de un archivo

load archivo variables

Ejemplo. Recuperar del archivo arch1 en el disco, las variables almacenadas

```
>> load arch1 a b
```

VERSION 3:

Acciones almacenadas en funciones con una variable global
Vector almacenado en un archivo en el disco
Uso de la instrucción try-catch-end para verificar

x: vector para almacenamiento de datos
ac: opción elegida por el usuario para control de acciones
datos: nombre del archivo en el disco para almacenar el vector

```
global x
try
   load datos x                %intenta cargar el vector x del archivo datos
catch
   x=[ ];                      %Si no está, inicia x como un vector nulo
end
ac=0;
while ac ~= 4
   menu1;
   ac=input('Elija una acción ');
   switch ac
      case 1, agregar;
      case 2, consultar;
      case 3, eliminar;
   end
end
save datos x
```

```
function menu1
clc;
disp('1) Agregar');
disp('2) Consultar');
disp('3) Eliminar');
disp('4) Salir');
```

```
function agregar
global x;
t=input('Ingrese dato ');
[e,p]=pertenece(t,x);
if e==0
   x=[x, t];
else
   disp('Ya existe este dato ');
   pause;
end
```

```
function consultar
global x;
t=input('Ingrese dato ');
[e,p]=pertenece(t,x);
if e==0
    disp('No está almacenado');
    pause;
else
    disp('Si está almacenado');
    pause;
end
```

```
function eliminar
global x;
t=input('Ingrese dato ');
[e,p]=pertenece(t,x);
if e==0
    disp('El dato no está almacenado ');
    pause;
else
    x(p)=[ ];
end
```

```
function [r,p]=pertenece(e,v)
n=length(v);
r=0;
p=0;
for i=1:n
   if e==v(i)
      r=1;
      p=i;
   end
end
```

11 Manejo de registros en MATLAB

Un registro o estructura es un dispositivo para contener datos cuyos componentes pueden ser de diferente tipo, a diferencia de los vectores y matrices cuyos elementos deben ser todos del mismo tipo.

Notación para manejo de los componentes de un **registro**:

Nombre del registro . nombre del componente

Ejemplo. Almacenar en un registro los datos de un estudiante:

e: **nombre del registros**
cod: **código del estudiante**
nom: **nombre del estudiante**
cursos: **lista de cursos tomados**

```
>> e.cod  = 125;
>> e.nom = 'Luis';
>> e.cursos    = [20, 30, 50, 40];
```

Mostrar el contenido del registro

```
>> e
ans =
        cod: 125
        nom: 'Luis'
        cursos: [20 30 50 40]
```

Notación para manejo de los componentes de un **vector (tabla) de registros**:

Nombre del vector(número de elemento) . nombre del componente

Ejemplo. Almacenar en el primer elemento de un vector, un registro con los datos de un estudiante:

e: **nombre del vector con registros**
cod: **código del estudiante**
nom; **nombre del estudiante**
cursos: **lista de cursos tomados**

```
>> e(1).cod  = 125;
>> e(1).nom = 'Luis';
>> e(1).cursos    = [20, 30, 50, 40];
```

Almacenar en el segundo elemento de vector, un registro con los datos de un nuevo estudiante:

```
>> e(2).cod  = 148;
>> e(2).nom = 'Maria';
>> e(2).cursos     = [20, 70, 80, 40];
```

Contenido actual del vector **e** con los datos de los dos estudiantes

e	**cod**	**nom**	**cursos**
1	125	'Luis'	20 30 40 50
2	148	'María'	20 70 80 40

Mostrar el contenido del segundo estudiante

```
>> e(2)
ans =
        cod: 148
        nom: 'Maria'
        cursos: [20 70 80 40]
```

Mostrar los cursos tomados por el estudiante 2

```
>> e(2).cursos
ans =
        20    70    80    40
```

Mostrar el tercer curso tomado por el estudiante 2

```
>> e(2).cursos(3)
ans
    80
```

También se puede definir previamente la estructura del registro antes de asignar los datos. Se deben indicar entre comillas los nombres de los componentes y asignar valores (nulos). Esta definición es opcional.

```
>> e=struct('cod',0,'nom','','cursos',[ ]);
>> e.cod(1)=125;
>> e.nom(1)='Luis';
>> e.cursos(1)=[20, 30, 50, 40];
```

Este vector de registros, si se almacena en un archivo en disco pudiera conceptualmente considerarse un archivo

Ejemplo. Almacenar en disco en un archivo con el nombre **arch** el vector **e** de registros.

```
>> save arch e
```

Si hay muchos datos, el ingreso conviene realizarlo mediante un programa.

Ejemplo. Desarrollo de un sistema con menú, funciones, registros y archivo en disco para control de ventas de los artículos de un almacén.

Variables

art: Nombre del vector de registros
arch: Nombre del archivo en disco

Datos de cada artículo

cod: código
cant: cantidad

NOTA. La función **ismember** no permite buscar elementos en un vector de registros, por lo cual se modificará nuestra conocida función **pertenece**

```
%Programa: Manejo de ventas de artículos
global art;
try
   load datos art   %Intenta cargar la variable art del archivo datos en el disco
catch
   art=[];          %Si no existe, inicia la variable art (vector)
end
ac=0;
while ac ~= 5
   menu1;
   ac=input('Elija una acción ');
   switch ac
      case 1, agregar;
      case 2, consultar;
      case 3, vender;
      case 4, eliminar;
   end
end
save datos art;   %guarda la variable art (vector) en el archivo datos
```

```
function menu1
clc;
disp('1) Agregar artículo');
disp('2) Consultar artículo');
disp('3) Vender artículo');
disp('4) Eliminar artículo');
disp('5) Salir');
```

```
function agregar
global art;
c=input('Ingrese código ');
[r,p]=pertenece1(c);
if r==0
  t.cod=c;
  t.cant=input('Ingrese cantidad ');
  art=[art, t];
else
  disp('Ya existe este dato ');
  pause;
end
```

```
function consultar
global art;
c=input('Ingrese código ');
[r,p]=pertenece1(c);
if r==0
   disp('No está almacenado');
   pause;
else
   disp('Cantidad disponible');
   disp(art(p).cant);
   pause;
end
```

```
function vender
global art;
c=input('Ingrese código ');
[r,p]=pertenece1(c);
if r==0
  disp('Dato no existe');
  pause;
else
  k=input('Ingrese cantidad ');
  if art(p).cant<k
     disp('Cantidad insuficiente');
     pause
  else
     t.cod=c;
     t.cant=art(p).cant-k;
     art(p)=t;
  end
end
```

```
function eliminar
global art;
c=input('Ingrese código ');
[r,p]=pertenece1(c);
if r==0
   disp('El dato no está almacenado ');
   pause;
else
   art(p)=[ ];
end
```

```
function [r,p]=pertenece1(c)
global art;
r=0
p=0;
n=length(art);
for i=1: n
   if c==art(i).cod
      r=1;
      p=i;
      return;
   end
end
```

Ejemplo. Desarrollo de una aplicación con funciones, registros y archivos

Sistema para registro de socios en un club.
Datos de cada socio: código, género, edad

Opciones

1) Ingresar socio	(Ingresar datos del socio)
2) Borrar socio	(Eliminar datos del socio)
3) Consultar	(Buscar si el socio está registrado)
4) Salir	

Variables

s:	Vector conteniendo códigos de socios (Variable global)
archsoc:	Nombre externo del archivo en disco

Componentes de cada registro de socio

c	Código (número entero)
g	Género ('M' o 'F')
e	Edad (Número entero, edad en años)

NOTA. La función **ismember** no permite buscar elementos en un vector de registros, por lo cual se modificará nuestra conocida función **pertenece**

```
%Programa para manejo de los datos de los socios de un club
global s
try
   load archsoc s;     % nombre externo del archivo
catch
   s=[ ];
end
opc=0;
while opc~=4
   clc;
   menu1;
   opc=input('Elija una opción ');
   switch opc
      case 1,  ingr;
      case 2,  elim;
      case 3,  cons;
   end
end
save archsoc s;
```

```
function menu1
%Lista de opciones
disp('1) Ingresar socio');
disp('2) Eliminar socio');
disp('3) Consultar');
disp('4) Salir');
```

```
function ingr
%Ingreso de los datos de un registro al archivo
global s;
t.c=input('ingrese código ');
t.g=input('ingrese género ','s');
t.e=input('ingrese edad ');
[r,p]=pertenece2(x)
if r==1
   disp('código ya existe');
   pause;
else
   s=[s,t];
end
```

```
function cons
%Consulta de un registro, dado su código
global s;
x=input('Ingrese código ');
[r,p]=pertenece2(x)
if r==1
   disp('socio si existe');
   disp(s(p).g);
   disp(s(p).e);
   pause;
else
   disp('socio no existe');
   pause;
end
```

```
function elim
%Elimina un registro del archivo s
%dado un código
global s;
x=input('ingrese código ');
[r,p]=pertenece2(x)
if r==0
   disp('socio no existe');
   pause;
else
   s(p)=[ ];
end
```

```
function [r,p]=pertenece2(x)
global s;
r=0;
p=0;
n=length(s);
for i=1:n
   if x==s(i).c
      r=1;
      p=i;
      return;
   end
end
```

11.1 Ejercicios de desarrollo de programas de aplicación

1.- Escriba un programa para el control de la cantidad de **n** artículos de una empresa mediante **menu** y **switch** y funciones para realizar cada opción. Al inicio lea **n** y asigne cero a la cantidad de todos los artículos

1) Comprar
2) Vender
3) Mostrar
4) Salir

2.- Escriba un programa para control del registro de los estudiantes para un evento.
El sistema debe incluir las siguientes opciones en un menú:

1) Registrar estudiante
2) Eliminar estudiante
3) Consultar registro de estudiantes
4) Mostrar estudiantes registrados
5) Salir

Use la instrucción **switch** y funciones para instrumentar cada opción y variables globales

3.- Desarrolle una aplicación para registro de socios en un club.

Opciones
1) Ingresar soci (Ingresar el código del socio en un vector)
2) Consultar (Buscar si el código del socio está en el vector)
3) Borrar socio (Eliminar el código del socio del vector)
4) Salir

Use un vector para almacenar los códigos de socios (Variable global)

4.- Se tiene una lista de n códigos de artículos (números enteros) y la cantidad disponible de cada uno, y otra lista de m clientes (números enteros) junto con el código del artículo que desea (un solo artículo por cliente) y la cantidad requerida. Almacene ambas listas en vectores de registros y determine la cantidad total sobrante o faltante de cada artículo para atender las solicitudes de todos los clientes.

12 Bibliografía

The MathWorks, Inc. *Using MATLAB Computation, Visualization, Programming*, version 7