Introducción

Héctor G. González Padilla

Marzo 2014

Hoja de Ruta

- Problemas y algoritmos

Hoja de Ruta

- Problemas y algoritmos
- Construción de diagramas de flujo

Hoja de Ruta

- Problemas y algoritmos
- Construción de diagramas de flujo
- Programas

Hoja de Ruta

- Problemas y algoritmos
- Construción de diagramas de flujo
- Programas
- MATLAB

Hoja de Ruta

- Problemas y algoritmos
- Construción de diagramas de flujo
- Programas
- MATLAB
  - Introducción a la programación

Hoja de Ruta

- Problemas y algoritmos
- Construción de diagramas de flujo
- Programas
- MATLAB
  - Introducción a la programación
  - Números y operadores aritmétricos

Hoja de Ruta

- Problemas y algoritmos
- Construción de diagramas de flujo
- Programas
- MATLAB
  - Introducción a la programación
  - Números y operadores aritmétricos
  - Vectores y matrices

Hoja de Ruta

- Problemas y algoritmos
- Construción de diagramas de flujo
- Programas
- MATLAB
  - Introducción a la programación
  - Números y operadores aritmétricos
  - Vectores y matrices
  - Matrices especiales

Hoja de Ruta

- Problemas y algoritmos
- Construción de diagramas de flujo
- Programas
- MATLAB
  - Introducción a la programación
  - Números y operadores aritmétricos
  - Vectores y matrices
  - Matrices especiales
  - Controles de flujo

Hoja de Ruta

- Problemas y algoritmos
- Construción de diagramas de flujo
- Programas
- MATLAB
  - Introducción a la programación
  - Números y operadores aritmétricos
  - Vectores y matrices
  - Matrices especiales
  - Controles de flujo
    - Repeticiones usando “for”

Hoja de Ruta

- Problemas y algoritmos
- Construción de diagramas de flujo
- Programas
- MATLAB
  - Introducción a la programación
  - Números y operadores aritmétricos
  - Vectores y matrices
  - Matrices especiales
  - Controles de flujo
    - Repeticiones usando “for”
    - Repeticiones usando “while”

Hoja de Ruta

- Problemas y algoritmos
- Construción de diagramas de flujo
- Programas
- MATLAB
  - Introducción a la programación
  - Números y operadores aritmétricos
  - Vectores y matrices
  - Matrices especiales
  - Controles de flujo
    - Repeticiones usando “for”
    - Repeticiones usando “while”
    - Repeticiones usando “if-else-end”

Problemas y Algoritmos

- Algoritmo

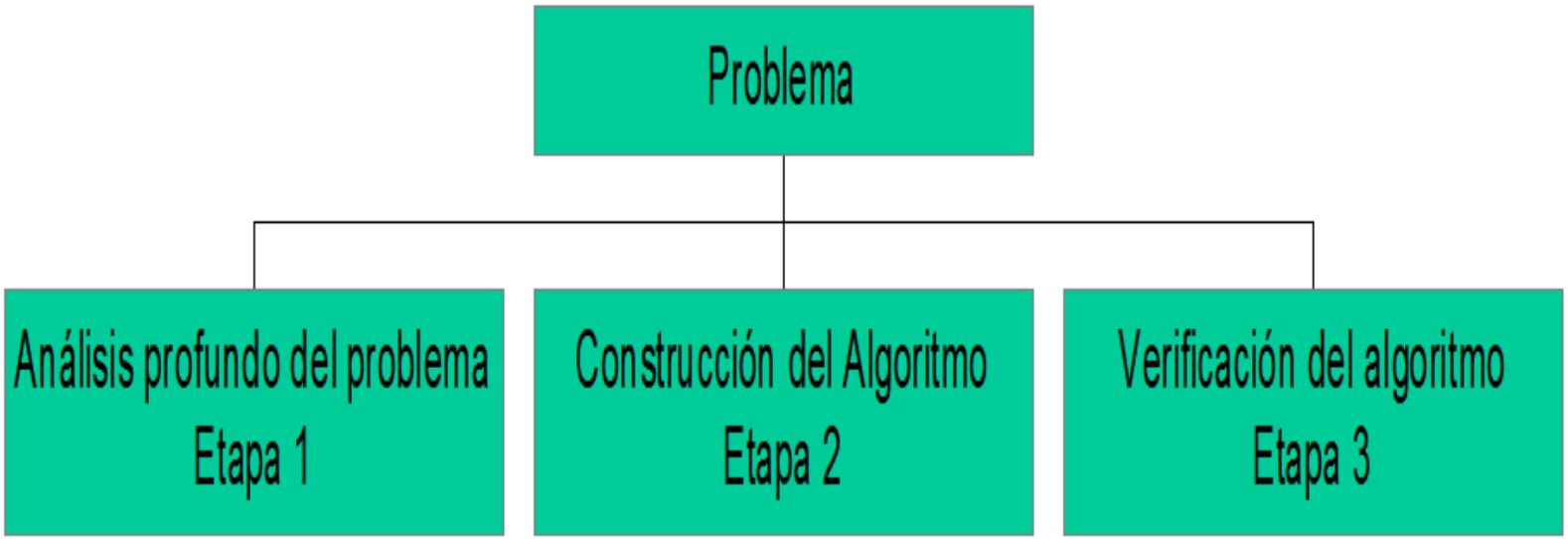
Problemas y Algoritmos

- Algoritmo
  - Un conjunto de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema.

Problemas y Algoritmos

- Algoritmo
  - Un conjunto de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema.
  - Ejemplo: preparar una receta de cocina.

Etapas de la Solución de un Problema



Etapas de la Solución de un Problema

- Problema

Etapas de la Solución de un Problema

- Problema
- Etapa 1 $\longrightarrow$ Análisis profundo del problema

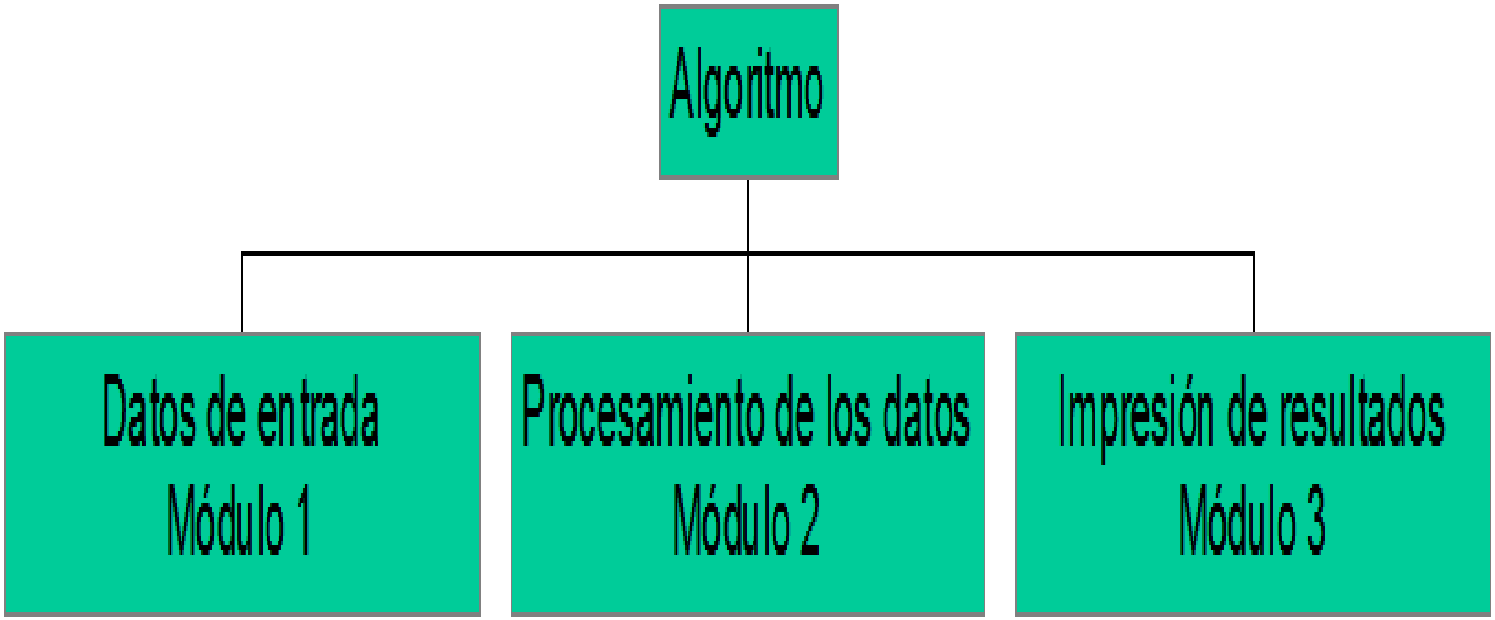
Etapas de la Solución de un Problema

- Problema
- Etapa 1 $\longrightarrow$ Análisis profundo del problema
- Etapa 2 $\longrightarrow$ Construcción del algoritmo

Etapas de la Solución de un Problema

- Problema
- Etapa 1 $\longrightarrow$ Análisis profundo del problema
- Etapa 2 $\longrightarrow$ Construcción del algoritmo
- Etapa 3 $\longrightarrow$ Verificación del algoritmo

Módulos de un Algoritmo



Módulos de un algoritmo

- Módulo 1 $\longrightarrow$ Datos de entrada $\longrightarrow$ representa la operación que permite el ingreso de los datos del problema.

Módulos de un algoritmo

- Módulo 1 $\longrightarrow$ Datos de entrada $\longrightarrow$ representa la operación que permite el ingreso de los datos del problema.
- Módulo 2 $\longrightarrow$ Procesamiento de los datos $\longrightarrow$ representa la operación o conjunto de operaciones secuenciales, cuyo objetivo es obtener la solución del problema.

Módulos de un algoritmo

- Módulo 1 $\longrightarrow$ Datos de entrada $\longrightarrow$ representa la operación que permite el ingreso de los datos del problema.
- Módulo 2 $\longrightarrow$ Procesamiento de los datos $\longrightarrow$ representa la operación o conjunto de operaciones secuenciales, cuyo objetivo es obtener la solución del problema.
- Módulo 3 $\longrightarrow$ Impresión de los datos $\longrightarrow$ representa una operación o conjunto de operaciones que permite comunicar al exterior el o los resultados obtenidos.

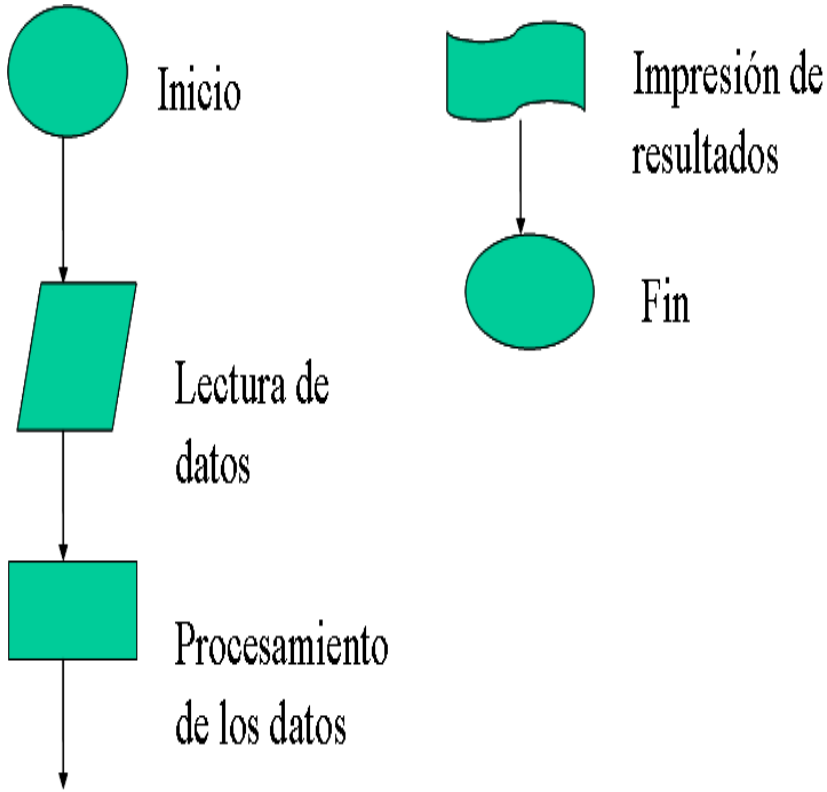
Diagramas de Flujo

- Un diagrama de flujo representa la esquematización gráfica de un algoritmo.

Diagramas de Flujo

- Un diagrama de flujo representa la esquematización gráfica de un algoritmo.
- Si el diagrama de flujo está completo y correcto, el paso del mismo a un lenguaje de programación es relativamente simple y directo.

Etapas en la Construcción de un Diagrama de Flujos



Etapas en la construcción de un Diagrama de Flujos

1. Inicio

Etapas en la construcción de un Diagrama de Flujos

1. Inicio
2. Lectura de datos

Etapas en la construcción de un Diagrama de Flujos

1. Inicio
2. Lectura de datos
3. Porcesamiento de datos

Etapas en la construcción de un Diagrama de Flujos

1. Inicio
2. Lectura de datos
3. Porcesamiento de datos
4. Impresión de resultados

Etapas en la construcción de un Diagrama de Flujos

1. Inicio
2. Lectura de datos
3. Porcesamiento de datos
4. Impresión de resultados
5. Fin

Identificadores, Constantes y Variables

- *Identificador:* nombre que se le da a las casillas de memoria.

Identificadores, Constantes y Variables

- *Identificador:* nombre que se le da a las casillas de memoria.
- *Constante:* son datos que no cambian durante la ejecución de un programa.

Identificadores, Constantes y Variables

- *Identificador:* nombre que se le da a las casillas de memoria.
- *Constante:* son datos que no cambian durante la ejecución de un programa.
- *Variables:* son objetos que pueden cambiar su valor durante la ejecución de un programa.

Construcción de un Diagrama de Flujo

- Construya un diagrama de flujo tal que dados los datos enteros A y B escriba el resultado de la siguiente oeración:

Construcción de un Diagrama de Flujo

- Construya un diagrama de flujo tal que dados los datos enteros A y B escriba el resultado de la siguiente oeración:
-

$$C = \left(\frac{A + B}{3} \right)^2$$

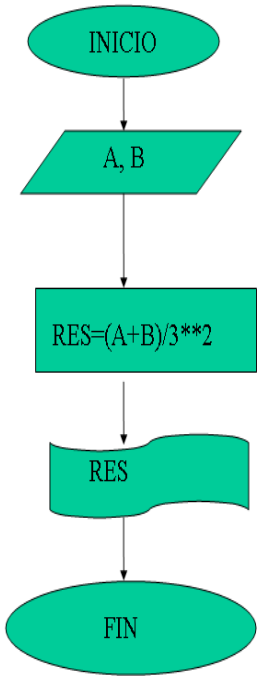
Construcción de un Diagrama de Flujo

- Construya un diagrama de flujo tal que dados los datos enteros A y B escriba el resultado de la siguiente oeración:
- $$C = \left(\frac{A + B}{3} \right)^2$$
- Datos: A, B.

Construcción de un Diagrama de Flujo

- Construya un diagrama de flujo tal que dados los datos enteros A y B escriba el resultado de la siguiente oeración:
- $$C = \left(\frac{A + B}{3} \right)^2$$
- Datos: A, B.
- A y B son variables de tipo entero que expresan los datos que se ingresan.

Diagrama de Flujo



Lectura de datos

El resultado de la expresión se almacena en la variable de tipo real RES

Impresión de la variables RES que almacena el resultado de la expresión

Ventana	Descripción
Command Window	Entradas de comandos a ser procesados por MATLAB
Command History	Listado de los comandos utilizados con anterioridad
Launch Pad	Listado de acceso a documentación, demos, etc.
Current Directory	Guía para la administración de archivos y directorios
Help	Guía para el acceso y visualización de documentación on-line
Workspace	Guía que permite acceder a variables de MATLAB
Array Editor	Guía que permite modificar el contenido de variables
Editor Debugger	Editor de textos para archivos de MATLAB

Cálculo Simple

- El nombre de las variables en MATLAB deben comenzar con una letra y puede ser seguido por letras y/o números hasta un máximo de 31 caracteres.

Cálculo Simple

- El nombre de las variables en MATLAB deben comenzar con una letra y puede ser seguido por letras y/o números hasta un máximo de 31 caracteres.

- $\gg s = 1 + 2$

Cálculo Simple

- El nombre de las variables en MATLAB deben comenzar con una letra y puede ser seguido por letras y/o números hasta un máximo de 31 caracteres.
- $\gg s = 1 + 2$
- $s = 3$

Cálculo Simple

- El nombre de las variables en MATLAB deben comenzar con una letra y puede ser seguido por letras y/o números hasta un máximo de 31 caracteres.
- $\gg s = 1 + 2$
- $s = 3$
- Para realizar un cálculo simple se debe ingresar el comando apropiado seguido de *Enter* o *Return*.

Números y Operaciones Aritmeticas

- Existen tres clases de números utilizados por MATLAB:

Números y Operaciones Aritmeticas

- Existen tres clases de números utilizados por MATLAB:
  - Enteros

Números y Operaciones Aritmeticas

- Existen tres clases de números utilizados por MATLAB:
  - Enteros
  - Reales

Números y Operaciones Aritmeticas

- Existen tres clases de números utilizados por MATLAB:
  - Enteros
  - Reales
  - Complejos

Operaciones básicas en MATLAB

Operación	Símbolo
Suma	+
Resta	-
Multiplicación	*
División	/ o \
Potenciación	^

Vectores

- Los elementos fundamentales que maneja MATLAB son vectores y matrices.

Vectores

- Los elementos fundamentales que maneja MATLAB son vectores y matrices.
- Vector: es un arreglo ordenado unidimensional de números.

Vectores

- Los elementos fundamentales que maneja MATLAB son vectores y matrices.
- Vector: es un arreglo ordenado unidimensional de números.
- El vector está delimitado por corchetes y sus elementos separados por espacios.

Vectores

- Los elementos fundamentales que maneja MATLAB son vectores y matrices.
- Vector: es un arreglo ordenado unidimensional de números.
- El vector está delimitado por corchetes y sus elementos separados por espacios.
- $\gg a = [1\ 2\ 3]$

Vectores

- Los elementos fundamentales que maneja MATLAB son vectores y matrices.
- Vector: es un arreglo ordenado unidimensional de números.
- El vector está delimitado por corchetes y sus elementos separados por espacios.
- $\gg a = [1\ 2\ 3]$
- $a =$

Vectores

- Los elementos fundamentales que maneja MATLAB son vectores y matrices.
- Vector: es un arreglo ordenado unidimensional de números.
- El vector está delimitado por corchetes y sus elementos separados por espacios.
- $\gg a = [1\ 2\ 3]$
- $a =$
- 1 2 3

Vectores

- Los elementos fundamentales que maneja MATLAB son vectores y matrices.
- Vector: es un arreglo ordenado unidimensional de números.
- El vector está delimitado por corchetes y sus elementos separados por espacios.
- $\gg a = [1\ 2\ 3]$
- $a =$
- 1 2 3
- Vectores columna se generan separando los elementos con punto y como (;)

Vectores

- Los elementos fundamentales que maneja MATLAB son vectores y matrices.
- Vector: es un arreglo ordenado unidimensional de números.
- El vector está delimitado por corchetes y sus elementos separados por espacios.
- $\gg a = [1\ 2\ 3]$
- $a =$
- 1 2 3
- Vectores columna se generan separando los elementos con punto y como (;)
- $\gg b = [1; 2; 3]$

Vectores

- Los elementos fundamentales que maneja MATLAB son vectores y matrices.
- Vector: es un arreglo ordenado unidimensional de números.
- El vector está delimitado por corchetes y sus elementos separados por espacios.
- $\gg a = [1\ 2\ 3]$
- $a =$
- 1 2 3
- Vectores columna se generan separando los elementos con punto y como (;)
- $\gg b = [1; 2; 3]$

  1
- 2

  3

Matrices

- Una matriz se define como

Matrices

- Una matriz se define como
- $\gg A = [1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9]$

Matrices

- Una matriz se define como
- $\gg A = [1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9]$
- $A =$

Matrices

- Una matriz se define como
- $\gg A = [1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9]$
- $A =$
- $\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$

Matrices

- Una matriz se define como
- $\gg A = [1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9]$
- $A =$
  $1 \quad 2 \quad 3$
- $4 \quad 5 \quad 6$
  $7 \quad 8 \quad 9$
- Las filas de una matriz están separadas por el símbolo ";" mientras que las columnas por un espacio.

Matrices Especiales

- La función *ones (m,n)* genera una matriz de m filas por n columnas que contiene sólo unos.

Matrices Especiales

- La función *ones (m,n)* genera una matriz de m filas por n columnas que contiene sólo unos.
- La función *zeros (m,n)* genera una matriz de m filas por n columnas que contiene sólo ceros.

Matrices Especiales

- La función *ones (m,n)* genera una matriz de m filas por n columnas que contiene sólo unos.
- La función *zeros (m,n)* genera una matriz de m filas por n columnas que contiene sólo ceros.
- la función *eye (n)* genera una matriz identidad de n x n.

Archivos con extensión .m

- Los comandos de MATLAB pueden organizarse de forma secuencial para formar un código (programa).

Archivos con extensión .m

- Los comandos de MATLAB pueden organizarse de forma secuencial para formar un código (programa).
- Los códigos se guardan en archivos con formato ASCII que poseen extensión *.m.

Archivos con extensión .m

- Los comandos de MATLAB pueden organizarse de forma secuencial para formar un código (programa).
- Los códigos se guardan en archivos con formato ASCII que poseen extensión *.m.
- Tipos de archivo m:

Archivos con extensión .m

- Los comandos de MATLAB pueden organizarse de forma secuencial para formar un código (programa).
- Los códigos se guardan en archivos con formato ASCII que poseen extensión *.m.
- Tipos de archivo m:
  - *Scripts:* contienen el cuerpo principal del programa.

Archivos con extensión .m

- Los comandos de MATLAB pueden organizarse de forma secuencial para formar un código (programa).
- Los códigos se guardan en archivos con formato ASCII que poseen extensión *.m.
- Tipos de archivo m:
  - *Scripts:* contienen el cuerpo principal del programa.
  - *Funciones*: pueden recibir variables como argumento y pueden entregar otras variables como resultado de las operaciones que realizan.

Ejemplo de Archivo Script

Cuadro: Archivo con Script

% archivo script grafico1.m	Comentarios
$x = pi/100 : pi/100 : 10 * pi$	Crea vector x
$y = \sin(x)./x;$	Evalúa la función deseada
plot(x,y)	Genera el gráfico
Grid	Dibuja en el gráfico la grilla punteada

Ejemplo de Función

- Función definida por el usuario cuyo argumento de entrada es un vector y la función devuelve un vector conteniendo los mismos elementos ordenados en forma descendiente.

Ejemplo de Función

- Función definida por el usuario cuyo argumento de entrada es un vector y la función devuelve un vector conteniendo los mismos elementos ordenados en forma descendiente.
- función [b] = descort(a)

Ejemplo de Función

- Función definida por el usuario cuyo argumento de entrada es un vector y la función devuelve un vector conteniendo los mismos elementos ordenados en forma descendiente.
- función [b] = descort(a)
- % función descort ordena en forma descendente un vector a.

Ejemplo de Función

- Función definida por el usuario cuyo argumento de entrada es un vector y la función devuelve un vector conteniendo los mismos elementos ordenados en forma descendiente.
- función [b] = descort(a)
- % función descort ordena en forma descendente un vector a.
- b = sort(-a);

Ejemplo de Función

- Función definida por el usuario cuyo argumento de entrada es un vector y la función devuelve un vector conteniendo los mismos elementos ordenados en forma descendiente.
- función [b] = descort(a)
- % función descort ordena en forma descendente un vector a.
- b = sort(-a);
- b = -b

Ejemplo de Función Continuación

Ejemplo numérico

$$\gg a = [pi - 10 \quad 35 \; 0{,}15]$$

$$a = [3{,}1416 - 10 \quad 35 \; 0{,}15]$$

$$\gg b = descsort(a)$$

$$b = 35 \; 3{,}1416 \; 0{,}15 \quad -10$$

Controles de Flujo

- Loops de tipo *for*

Controles de Flujo

- Loops de tipo *for*
- Loops de tipo *while*

Controles de Flujo

- Loops de tipo *for*
- Loops de tipo *while*
- Loops de tipo *if*

Controles de Flujo

- Loops de tipo *for*
- Loops de tipo *while*
- Loops de tipo *if*
- Loop de tipo if - else end

Controles de Flujo

- Loops de tipo *for*
- Loops de tipo *while*
- Loops de tipo *if*
- Loop de tipo if - else end
- Loop de tipo if - elseif end

Controles de Flujo

- Loops de tipo *for*
- Loops de tipo *while*
- Loops de tipo *if*
- Loop de tipo if - else end
- Loop de tipo if - elseif end
- Loops del tipo *switch-case*

Controles de Flujo

Loop de tipo for

- La sintaxis es la siguiente:

Controles de Flujo

Loop de tipo for

- La sintaxis es la siguiente:
- for i = vector

Controles de Flujo

Loop de tipo for

- La sintaxis es la siguiente:
- for i = vector
- comandos

Controles de Flujo

Loop de tipo for

- La sintaxis es la siguiente:
- for i = vector
- comandos
- end

Controles de Flujo

Loop de tipo for

- La sintaxis es la siguiente:
- for i = vector
- comandos
- end
- Los comandos que se encuentran entre las sentencias *for* y *end* son ejecutados para todos los valores dentro del vector.

Controles de Flujo

Ejemplo de loop de tipo for

- Calcular la función seno (x) en puntos equidistantes en n/10 dentro del intervalo [0,pi]

Controles de Flujo

Ejemplo de loop de tipo for

- Calcular la función seno (x) en puntos equidistantes en n/10 dentro del intervalo [0,pi]
- for n $= 0 : 10$

Controles de Flujo

Ejemplo de loop de tipo for

- Calcular la función seno (x) en puntos equidistantes en n/10 dentro del intervalo [0,pi]
- for n = 0 : 10
- x(n+1) = sin(pi*n/10);

Controles de Flujo

Ejemplo de loop de tipo for

- Calcular la función seno (x) en puntos equidistantes en n/10 dentro del intervalo [0,pi]
- for n = 0 : 10
- x(n+1) = sin(pi*n/10);
- end

Controles de Flujo

Ejemplo de loop de tipo for

- Calcular la función seno (x) en puntos equidistantes en n/10 dentro del intervalo [0,pi]
- for n = 0 : 10
- x(n+1) = sin(pi*n/10);
- end
- disp(x)

Controles de Flujo

Loop de tipo if

- Este tipo de enunciado se utiliza para obtener un resultado si se satisface una condición en una expresión.

Controles de Flujo

Loop de tipo if

- Este tipo de enunciado se utiliza para obtener un resultado si se satisface una condición en una expresión.
- La sintaxis es:

Controles de Flujo

Loop de tipo if

- Este tipo de enunciado se utiliza para obtener un resultado si se satisface una condición en una expresión.
- La sintaxis es:
  - If condición, expresión

Controles de Flujo

Loop de tipo if

- Este tipo de enunciado se utiliza para obtener un resultado si se satisface una condición en una expresión.
- La sintaxis es:
  - If condición, expresión
  - end

Controles de Flujo

Ejemplo de loop de tipo if

- Cálculo de un volumen

Controles de Flujo

Ejemplo de loop de tipo if

- Cálculo de un volumen
- r = 2;

Controles de Flujo

Ejemplo de loop de tipo if

- Cálculo de un volumen
- r = 2;
- if r > 0, vol = (4/3)*pi*r ^3;

Controles de Flujo

Ejemplo de loop de tipo if

- Cálculo de un volumen
- r = 2;
- if r > 0, vol = (4/3)*pi*r ^3;
- disp(vol)

Controles de Flujo

Ejemplo de loop de tipo if

- Cálculo de un volumen
- r = 2;
- if r > 0, vol = (4/3)*pi*r ^3;
- disp(vol)
- end

Controles de Flujo

Loop de tipo while

- Este tipo de loop se utiliza cuando se desconoce el número de iteraciones que son necesarias y por el contrario lo que se tiene es una condición a satisfacer.

Controles de Flujo

Loop de tipo while

- Este tipo de loop se utiliza cuando se desconoce el número de iteraciones que son necesarias y por el contrario lo que se tiene es una condición a satisfacer.
- La sintaxis es:

Controles de Flujo

Loop de tipo while

- Este tipo de loop se utiliza cuando se desconoce el número de iteraciones que son necesarias y por el contrario lo que se tiene es una condición a satisfacer.
- La sintaxis es:
- while condición

Controles de Flujo

Loop de tipo while

- Este tipo de loop se utiliza cuando se desconoce el número de iteraciones que son necesarias y por el contrario lo que se tiene es una condición a satisfacer.
- La sintaxis es:
- while condición
- comandos

Controles de Flujo

Loop de tipo while

- Este tipo de loop se utiliza cuando se desconoce el número de iteraciones que son necesarias y por el contrario lo que se tiene es una condición a satisfacer.
- La sintaxis es:
- while condición
- comandos
- end

Controles de Flujo

Ejemplo de loop de tipo while

- Partiendo de un número inicial (pi) lo dividimos sucesivamente por dos y queremos saber cuál es el menor número de esta sucesión que sea mayor a 0.01.

Controles de Flujo

Ejemplo de loop de tipo while

- Partiendo de un número inicial (pi) lo dividimos sucesivamente por dos y queremos saber cuál es el menor número de esta sucesión que sea mayor a 0.01.
- q = pi

Controles de Flujo

Ejemplo de loop de tipo while

- Partiendo de un número inicial (pi) lo dividimos sucesivamente por dos y queremos saber cuál es el menor número de esta sucesión que sea mayor a 0.01.
- q = pi
- while(q/2 > 0.01)

Controles de Flujo

Ejemplo de loop de tipo while

- Partiendo de un número inicial (pi) lo dividimos sucesivamente por dos y queremos saber cuál es el menor número de esta sucesión que sea mayor a 0.01.
- q = pi
- while(q/2 > 0.01)
- q = q/2

Controles de Flujo

Ejemplo de loop de tipo while

- Partiendo de un número inicial (pi) lo dividimos sucesivamente por dos y queremos saber cuál es el menor número de esta sucesión que sea mayor a 0.01.
- q = pi
- while(q/2 > 0.01)
- q = q/2
- end

Controles de Flujo

Ejemplo de loop de tipo while

- Partiendo de un número inicial (pi) lo dividimos sucesivamente por dos y queremos saber cuál es el menor número de esta sucesión que sea mayor a 0.01.
- q = pi
- while(q/2 > 0.01)
- q = q/2
- end
- disp(q)

Controles de Flujo

Loop de tipo if - else end

- Esta construcción se utiliza para identificar una, dos o más alternativas.

Controles de Flujo

Loop de tipo if - else end

- Esta construcción se utiliza para identificar una, dos o más alternativas.
- La sintaxis es:

Controles de Flujo

Loop de tipo if - else end

- Esta construcción se utiliza para identificar una, dos o más alternativas.
- La sintaxis es:
- if expresión

Controles de Flujo

Loop de tipo if - else end

- Esta construcción se utiliza para identificar una, dos o más alternativas.
- La sintaxis es:
- if expresión
- comandos

Controles de Flujo

Loop de tipo if - else end

- Esta construcción se utiliza para identificar una, dos o más alternativas.
- La sintaxis es:
- if expresión
- comandos
- end

Controles de Flujo

Loop de tipo if - else end

- Esta construcción se utiliza para identificar una, dos o más alternativas.
- La sintaxis es:
- if expresión
- comandos
- end
- if expresión

Controles de Flujo

Loop de tipo if - else end

- Esta construcción se utiliza para identificar una, dos o más alternativas.
- La sintaxis es:
- if expresión
- comandos
- end
- if expresión
- comandos

Controles de Flujo

Loop de tipo if - else end

- Esta construcción se utiliza para identificar una, dos o más alternativas.
- La sintaxis es:
- if expresión
- comandos
- end
- if expresión
- comandos
- else

Controles de Flujo

Loop de tipo if - else end

- Esta construcción se utiliza para identificar una, dos o más alternativas.
- La sintaxis es:
- if expresión
- comandos
- end
- if expresión
- comandos
- else
- comandos

Controles de Flujo

Loop de tipo if - else end

- Esta construcción se utiliza para identificar una, dos o más alternativas.
- La sintaxis es:
- if expresión
- comandos
- end
- if expresión
- comandos
- else
- comandos
- end

Controles de Flujo

Ejemplo de Loop de tipo if - else end

- x = -33

Controles de Flujo

Ejemplo de Loop de tipo if - else end

- x = -33
- if x > 0

Controles de Flujo

Ejemplo de Loop de tipo if - else end

- x = -33
- if x $>$ 0
- y = log(x)

Controles de Flujo

Ejemplo de Loop de tipo if - else end

- x = -33
- if x > 0
- y = log(x)
- else

Controles de Flujo

Ejemplo de Loop de tipo if - else end

- x = -33
- if x > 0
- y = log(x)
- else
- x = 'número negativo'

Controles de Flujo

Ejemplo de Loop de tipo if - else end

- x = -33
- if x > 0
- y = log(x)
- else
- x = 'número negativo'
- end

Controles de Flujo

Loop de tipo if - elseif end

- Es un control de flujo más poderoso.

Controles de Flujo

Loop de tipo if - elseif end

- Es un control de flujo más poderoso.
- La sintaxis es:

Controles de Flujo

Loop de tipo if - elseif end

- Es un control de flujo más poderoso.
- La sintaxis es:
- if condición1

Controles de Flujo

Loop de tipo if - elseif end

- Es un control de flujo más poderoso.
- La sintaxis es:
- if condición1
- comandos

Controles de Flujo

Loop de tipo if - elseif end

- Es un control de flujo más poderoso.
- La sintaxis es:
- if condición1
- comandos
- elseif condición2

Controles de Flujo

Loop de tipo if - elseif end

- Es un control de flujo más poderoso.
- La sintaxis es:
- if condición1
- comandos
- elseif condición2
- comandos

Controles de Flujo

Loop de tipo if - elseif end

- Es un control de flujo más poderoso.
- La sintaxis es:
- if condición1
- comandos
- elseif condición2
- comandos
- elseif ...

Controles de Flujo

Loop de tipo if - elseif end

- Es un control de flujo más poderoso.
- La sintaxis es:
- if condición1
- comandos
- elseif condición2
- comandos
- elseif ...
- ...

Controles de Flujo

Loop de tipo if - elseif end

- Es un control de flujo más poderoso.
- La sintaxis es:
- if condición1
- comandos
- elseif condición2
- comandos
- elseif ...
- ...
- else

Controles de Flujo

Loop de tipo if - elseif end

- Es un control de flujo más poderoso.
- La sintaxis es:
- if condición1
- comandos
- elseif condición2
- comandos
- elseif ...
- ...
- else
- comandos

Controles de Flujo

Loop de tipo if - elseif end

- Es un control de flujo más poderoso.
- La sintaxis es:
- if condición1
- comandos
- elseif condición2
- comandos
- elseif ...
- ...
- else
- comandos
- end

Controles de Flujo

Ejemplo de Loop de tipo if - elseif end

- x = '33';

Controles de Flujo

Ejemplo de Loop de tipo if - elseif end

- x = '33';
- if ~isnumeric(x)

Controles de Flujo

Ejemplo de Loop de tipo if - elseif end

- x = '33';
- if ~isnumeric(x)
- x = 'No es un número'

Controles de Flujo

Ejemplo de Loop de tipo if - elseif end

- x = '33';
- if ~isnumeric(x)
- x = 'No es un número'
- elseif isnumeric(x) & x < 0

Controles de Flujo

Ejemplo de Loop de tipo if - elseif end

- x = '33';
- if ~isnumeric(x)
- x = 'No es un número'
- elseif isnumeric(x) & x < 0
- x = 'Número negativo'

Controles de Flujo

Ejemplo de Loop de tipo if - elseif end

- x = '33';
- if ~isnumeric(x)
- x = 'No es un número'
- elseif isnumeric(x) & x < 0
- x = 'Número negativo'
- else

Controles de Flujo

Ejemplo de Loop de tipo if - elseif end

- x = '33';
- if ~isnumeric(x)
- x = 'No es un número'
- elseif isnumeric(x) & x < 0
- x = 'Número negativo'
- else
- x = log(x)

Controles de Flujo

Ejemplo de Loop de tipo if - elseif end

- x = '33';
- if ~isnumeric(x)
- x = 'No es un número'
- elseif isnumeric(x) & x < 0
- x = 'Número negativo'
- else
- x = log(x)
- end

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.
- La sintaxis es :

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.
- La sintaxis es :
- switch condition

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.
- La sintaxis es :
- switch condition
- case condición1

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.
- La sintaxis es :
- switch condition
- case condición1
- comandos

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.
- La sintaxis es :
- switch condition
- case condición1
- comandos
- case condición2

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.
- La sintaxis es :
- switch condition
- case condición1
- comandos
- case condición2
- comandos

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.
- La sintaxis es :
- switch condition
- case condición1
- comandos
- case condición2
- comandos
- case ...

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.
- La sintaxis es :
- switch condition
- case condición1
- comandos
- case condición2
- comandos
- case ...
- ...

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.
- La sintaxis es :
- switch condition
- case condición1
- comandos
- case condición2
- comandos
- case ...
- ...
- otherwise

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.
- La sintaxis es :
- switch condition
- case condición1
- comandos
- case condición2
- comandos
- case ...
- ...
- otherwise
- comandos

Controles de Flujo

Loop de tipo switch

- El comando *switch* es preferible si hay varias alternativas que puden ser expresadas como casos particulares de una condición comun.
- La sintaxis es :
- switch condition
- case condición1
- comandos
- case condición2
- comandos
- case ...
- ...
- otherwise
- comandos
- end

Relaciones

Las comparaciones en MATLAB se realizan utilizando los siguientes operadores

Cuadro: Relaciones

Operador	Descripción
<	Menor que
<=	Menor o igual a
>	Mayor que
>=	Mayor o igual a
==	Igual a
.=	Distinto de

Operadores Lógicos

Cuadro: Operadores Lgicos

Operador Lógico	Descripción
&	And
|	Or
~	Not

Bibliografía

- Lepone, Francisco. Introducción a la programación en MATLAB.

Bibliografía

- Lepone, Francisco. Introducción a la programación en MATLAB.
- Nakamura, S. Análisis Numérico y Visualización Gráfica con MATLAB. Pearson Education.

Bibliografía

- Lepone, Francisco. Introducción a la programación en MATLAB.
- Nakamura, S. Análisis Numérico y Visualización Gráfica con MATLAB. Pearson Education.
- Pérez, C. MATLAB y sus aplicaciones en las ciencias y la Ingeniería. Pearson-Prentice-Hall.