UNIVERSIDAD DE COSTA RICA

Programando con MATLAB

Mario De León Urbina

Escuela de Matemática

13 de agosto de 2023

0 Contenidos

UNIVERSIDAD DE COSTA RICA

1 Contenidos

1 MATLAB

1 MATLAB

Lenguaje

1 MATLAB

Lenguaje

`MATLAB` posee su propio lenguaje de programación el cual permite programar aplicaciones, principalmente numéricas. Es un lenguaje de alto nivel que no es tan abarcante como otros lenguajes, pero las instrucciones no son tan complicadas.

1 MATLAB

Lenguaje

`MATLAB` posee su propio lenguaje de programación el cual permite programar aplicaciones, principalmente numéricas. Es un lenguaje de alto nivel que no es tan abarcante como otros lenguajes, pero las instrucciones no son tan complicadas.

- Combina un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y arrays directamente.

1 MATLAB

Lenguaje

`MATLAB` posee su propio lenguaje de programación el cual permite programar aplicaciones, principalmente numéricas. Es un lenguaje de alto nivel que no es tan abarcante como otros lenguajes, pero las instrucciones no son tan complicadas.

- Combina un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y arrays directamente.

  > Observe el video ¿Qué es MATLAB?

Lenguaje

`MATLAB` posee su propio lenguaje de programación el cual permite programar aplicaciones, principalmente numéricas. Es un lenguaje de alto nivel que no es tan abarcante como otros lenguajes, pero las instrucciones no son tan complicadas.

- Combina un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y arrays directamente.
  - Observe el video ¿Qué es MATLAB?
  - Revise el contenido de Wikipedia sobre MATLAB

1 MATLAB

1 MATLAB

Análisis de datos

Explore, modele y visualice datos

Gráficas

Visualice y explore datos

Desarrollo de algoritmos

Diseñe algoritmos para aplicaciones de escritorio y embebidas

Creación de apps

Cree apps web y de escritorio

Uso de MATLAB con otros lenguajes

Utilice MATLAB con Python, C/C+, Fortran, Java y otros lenguajes

Hardware

Conecte MATLAB con hardware

Cálculo paralelo

Efectúe cálculos a gran escala mediante equipos multinúcleo, GPU, clusters y nubes

Despliegue en escritorio y web

Comparta sus programas de MATLAB

Cálculo en la nube

Realice la ejecución en entornos de nube, desde MathWorks Cloud hasta nubes públicas como AWS y Azure

2 Contenidos

2 Funciones lógicas

- Comparaciones:

`A == B, A ~= B, A <= B, A >= B, A < B, A > B`

comparaciones componente a componente de matrices;
1 es para 'verdadero' y 0 para 'falso'.

- Comparaciones:

`A == B, A ~= B, A <= B, A >= B, A < B, A > B`

comparaciones componente a componente de matrices;
1 es para 'verdadero' y 0 para 'falso'.

- Operaciones lógicas booleanas:

`a && b, a || b`
son los operadores lógicos 'y' y 'o', respectivamente.

$\sim$ *expresión*, `not` (*expresión*)
ambas retornan la negación lógica de la expresión.

3 Contenidos

3 Pseudocódigo

3 Pseudocódigo

Componentes

3 Pseudocódigo

Componentes

Un pseudocódigo puede dividirse en 5 componentes:

Componentes

Un pseudocódigo puede dividirse en 5 componentes:

- Variables

3 Pseudocódigo

Componentes

Un pseudocódigo puede dividirse en 5 componentes:

- Variables
- Asignaciones

Componentes

Un pseudocódigo puede dividirse en 5 componentes:

- Variables
- Asignaciones
- Input/output

3 Pseudocódigo

Componentes

Un pseudocódigo puede dividirse en 5 componentes:

- Variables
- Asignaciones
- Input/output
- Selección

3 Pseudocódigo

Componentes

Un pseudocódigo puede dividirse en 5 componentes:

- Variables
- Asignaciones
- Input/output
- Selección
- Repetición

3 Variables

3 Variables

Variable

3 Variables

Variable

- Una variable tiene un nombre, un tipo de datos y un valor.

3 Variables

Variable

- Una variable tiene un nombre, un tipo de datos y un valor.
- Hay una ubicación en la memoria asociada con cada variable.

3 Variables

Variable

- Una variable tiene un nombre, un tipo de datos y un valor.
- Hay una ubicación en la memoria asociada con cada variable.
- Una variable se puede llamar cualquier cosa o se le puede dar cualquier nombre.

3 Variables

Variable

- Una variable tiene un nombre, un tipo de datos y un valor.
- Hay una ubicación en la memoria asociada con cada variable.
- Una variable se puede llamar cualquier cosa o se le puede dar cualquier nombre.
- Se considera una buena práctica usar nombres de variables que sean relevantes para la tarea a mano.

3 Asignaciones

3 Asignaciones

Assignment

Assignment

- La asignación es el acto físico de colocar un valor en una variable. La asignación se puede mostrar usando

Assignment

- La asignación es el acto físico de colocar un valor en una variable. La asignación se puede mostrar usando

  $\text{set} = 5;$

Assignment

- La asignación es el acto físico de colocar un valor en una variable. La asignación se puede mostrar usando

  $\text{set} = 5;$

  $\text{set} = \text{num} + \text{set};$

Assignment

- La asignación es el acto físico de colocar un valor en una variable. La asignación se puede mostrar usando

  $\text{set} = 5;$

  $\text{set} = \text{num} + \text{set};$

- El lado izquierdo es la variable en la que se almacena un valor y el lado derecho es donde se accede a la variable.

Assignment

- La asignación es el acto físico de colocar un valor en una variable. La asignación se puede mostrar usando

  $\text{set} = 5;$

  $\text{set} = \text{num} + \text{set};$
- El lado izquierdo es la variable en la que se almacena un valor y el lado derecho es donde se accede a la variable.
- Cuando una se le asigna un valor a la variable, el valor anterior se sobrescribe con el valor nuevo, por lo que el valor anterior desaparece. $x = 5$ no significa que x es igual a 5; significa establecer la variable x para que tenga el valor 5. Dar a x el valor 5, hace que x sea igual a 5.

3 Input / Output

Input-Output

Input-Output

- Ambos tratan con una fuente externa (puede ser un usuario u otro programa) que recibe o da información.

Input-Output

- Ambos tratan con una fuente externa (puede ser un usuario u otro programa) que recibe o da información.
  - Un ejemplo sería asumir que un restaurante de comida rápida es un programa. Un conductor (usuario) enviaría su pedido de una hamburguesa y papas fritas (entrada), luego conduciría hasta la ventana lateral y recogería la comida ordenada (salida).

Input-Output

- Ambos tratan con una fuente externa (puede ser un usuario u otro programa) que recibe o da información.
  - Un ejemplo sería asumir que un restaurante de comida rápida es un programa. Un conductor (usuario) enviaría su pedido de una hamburguesa y papas fritas (entrada), luego conduciría hasta la ventana lateral y recogería la comida ordenada (salida).
  - Input – Read / get / input

Input-Output

- Ambos tratan con una fuente externa (puede ser un usuario u otro programa) que recibe o da información.
  - Un ejemplo sería asumir que un restaurante de comida rápida es un programa. Un conductor (usuario) enviaría su pedido de una hamburguesa y papas fritas (entrada), luego conduciría hasta la ventana lateral y recogería la comida ordenada (salida).
  - Input – Read / get / input
  - Output – Write / display / print

3 Selección

3 Selección

Selección

Selección

- La construcción de selección permite elegir entre realizar una acción y omitirla. Son nuestras declaraciones condicionales. Las declaraciones de selección se escriben así:

Selección

- La construcción de selección permite elegir entre realizar una acción y omitirla. Son nuestras declaraciones condicionales. Las declaraciones de selección se escriben así:

  if (conditional statement)

  statement list

  else

  statement list

3 Repetición

3 Repetición

Repetición

Repetición

- La repetición es una construcción que permite que las instrucciones se ejecuten varias veces (IE repetidas). En un problema de repetición

Repetición

- La repetición es una construcción que permite que las instrucciones se ejecuten varias veces (IE repetidas). En un problema de repetición
  - Se inicializa el conteo

Repetición

- La repetición es una construcción que permite que las instrucciones se ejecuten varias veces (IE repetidas). En un problema de repetición
  - Se inicializa el conteo
  - Probado

Repetición

- La repetición es una construcción que permite que las instrucciones se ejecuten varias veces (IE repetidas). En un problema de repetición
  - Se inicializa el conteo
  - Probado
  - Incrementado

Repetición

- La repetición es una construcción que permite que las instrucciones se ejecuten varias veces (IE repetidas). En un problema de repetición
  - Se inicializa el conteo
  - Probado
  - Incrementado
- Los problemas de repetición se muestran como:

Repetición

- La repetición es una construcción que permite que las instrucciones se ejecuten varias veces (IE repetidas). En un problema de repetición
  - Se inicializa el conteo
  - Probado
  - Incrementado
- Los problemas de repetición se muestran como:

  while (condition statement)

  statement list

Algorithm 1 Método de bisección

Require: f continua, intervalo inicial $I_0 \leftarrow [a, b]$ con $f(a)f(b) < 0$
Ensure: Aproximación c_k de una raíz c de f en I_0

```
k ← 0;
while la iteración no haya convergido do
    c_k ← (a_k + b_k)/2;
    if c_k es una raíz then
        Retorne c_k;
    end if
    if f(c_k)f(b_k) < 0 then
        a_{k+1} ← c_k, b_{k+1} ← b_k;
    else
        a_{k+1} ← a_k, b_{k+1} ← c_k;
    end if
    k ← k + 1;
end while
```

```
function [c] = Biseccion(f,a,b,tol)
if f(a)*f(b)>0
    error('No se cumple el cambio de signo');
end
k=0;
c=(a+b)/2;
while abs(f(c))> tol
    if f(c)<0 && f(a)<0
        a=c;
    else
        b=c;
    end
    c=(a+b)/2;
    k=k+1;
end
end
```

4 Contenidos

4 Condicionales

4 Condicionales

4 Condicionales


```
if expresión
    instrucciones
elseif expresión
    instrucciones
else
    instrucciones
end
```

4 Condicionales


```
if expresión
    instrucciones
elseif expresión
    instrucciones
else
    instrucciones
end
```

Las instrucciones son ejecutadas si las partes reales de todas las entradas de la (matriz evaluada) expresión son no-cero; ‘`elseif`’ y ‘`else`’ son opcionales y múltiples ‘`elseif`’ pueden utilizarse.

if-end

if-end

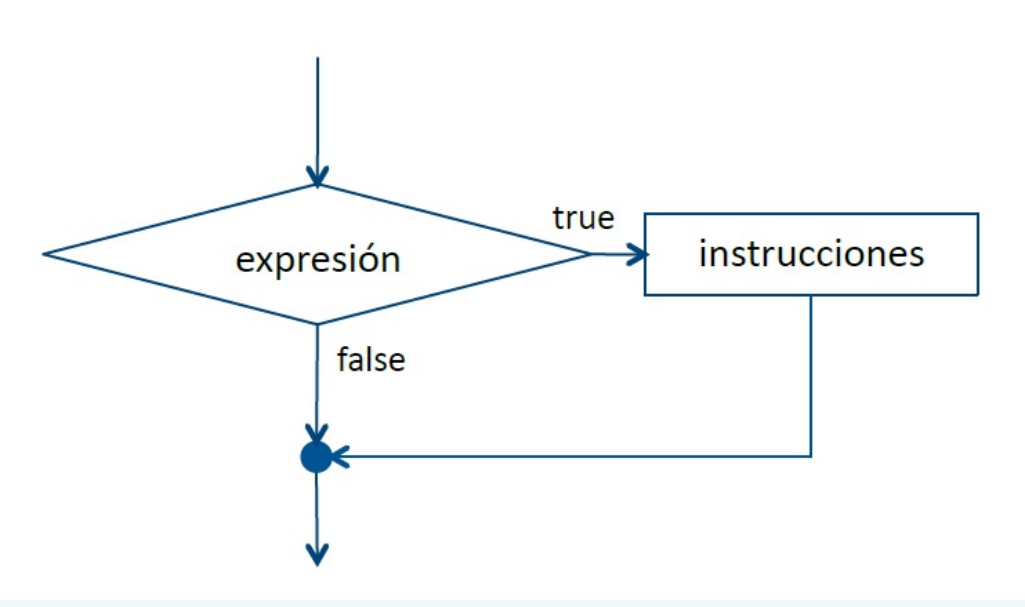


Figura: Diagrama de flujo de la estructura condicional simple.

if-else-end

if-else-end

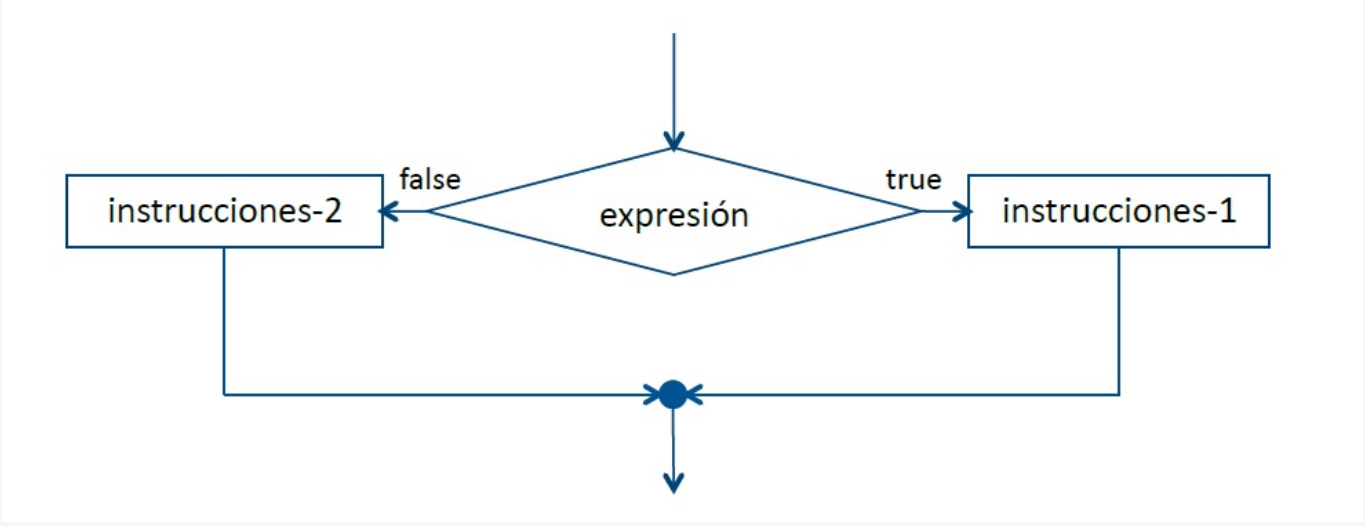
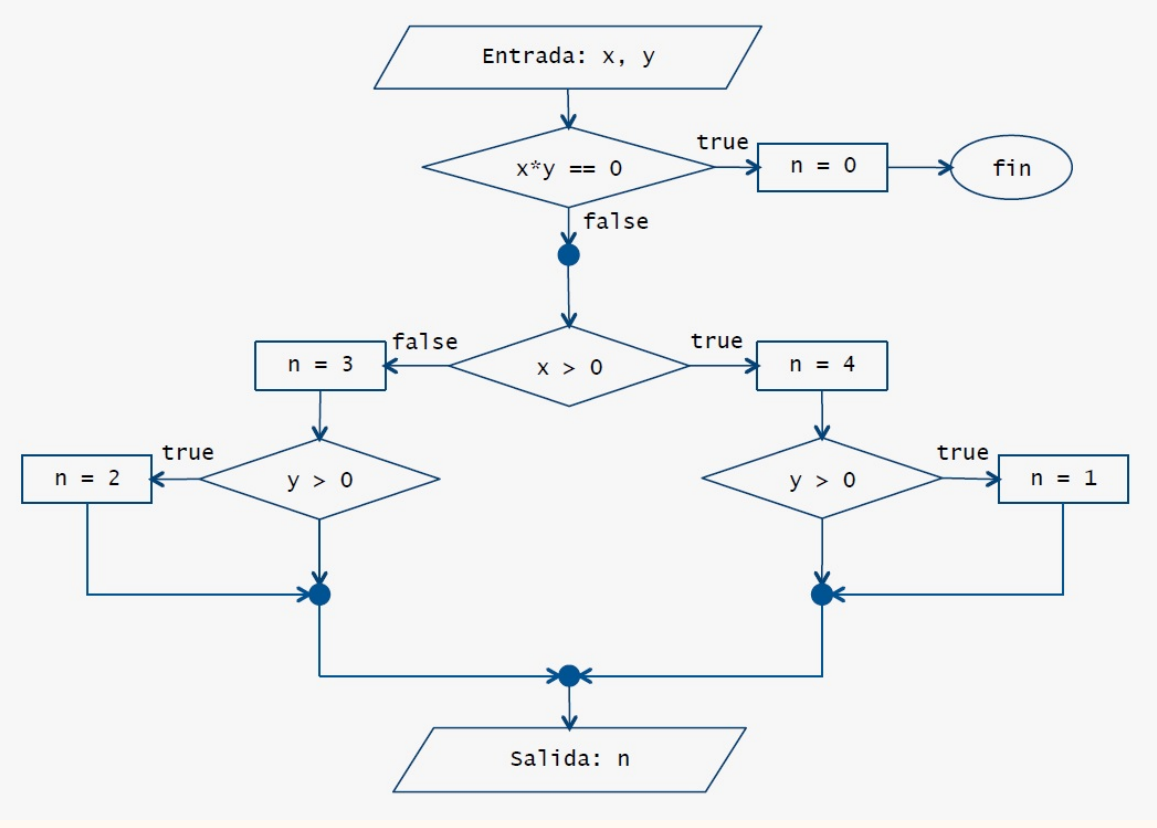


Figura: Diagrama de flujo de la estructura condicional doble.

4 Condicionales

Ejemplo: ubicando el par (x, y) en el plano cartesiano

Ejemplo: ubicando el par (x, y) en el plano cartesiano



if-elseif-else-end

if-elseif-else-end

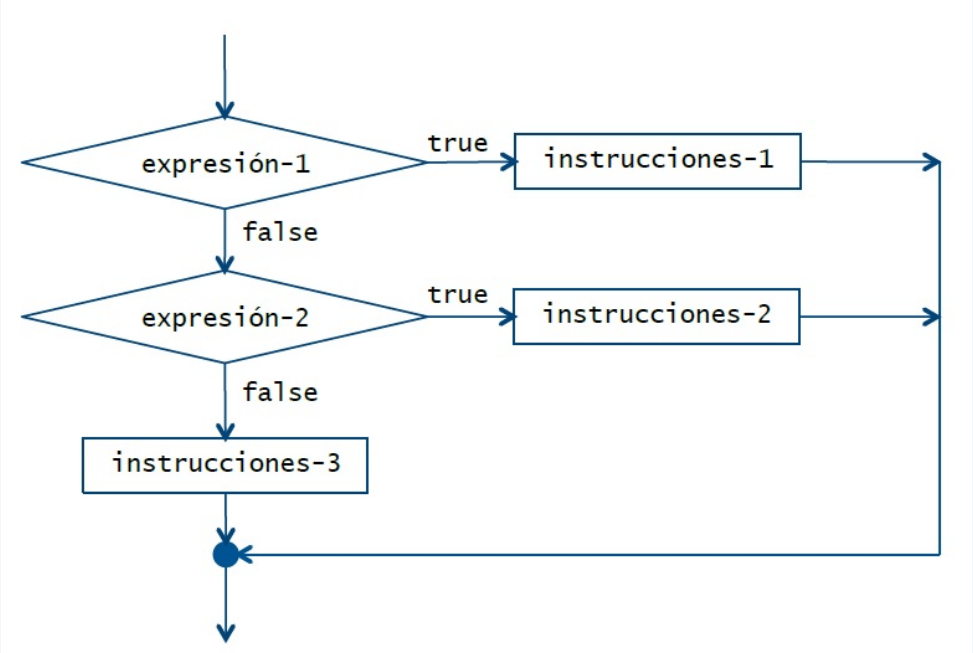


Figura: Diagrama de flujo de la estructura condicional múltiple.

4 Ciclos `for`:

4 Ciclos `for`:

4 Ciclos `for`:


```
for Variable = vector
    instrucciones
end
```

4 Ciclos `for`:

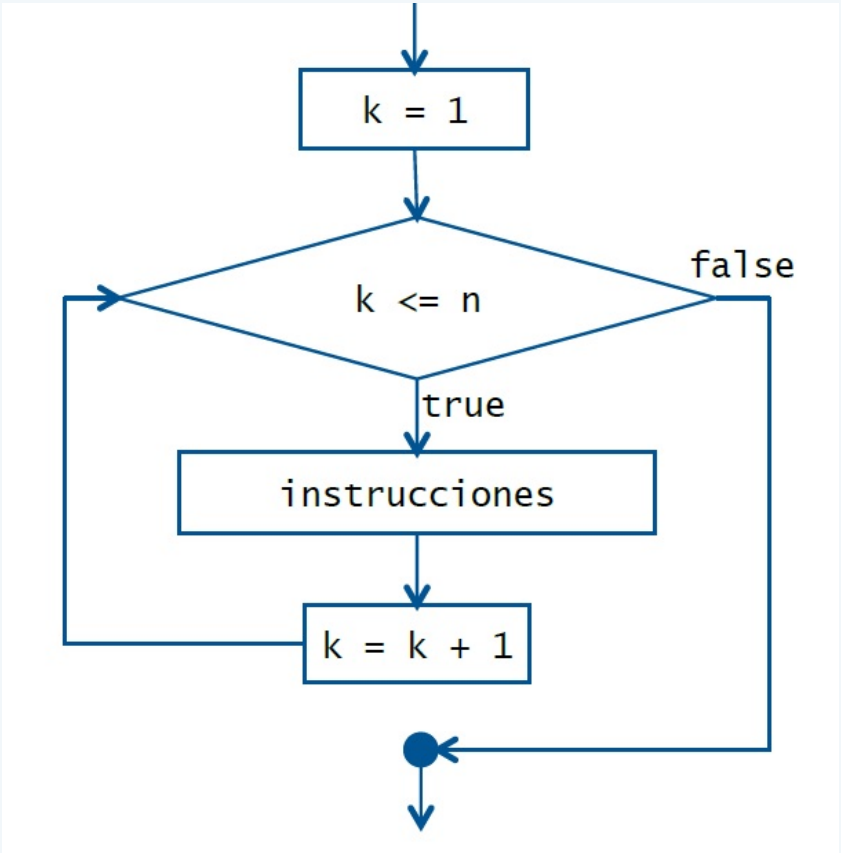
`for` *Variable* = *vector*
    instrucciones
`end`

ejecuta instrucciones varias veces donde a la variable se le asigna sucesivamente el valor de las componentes del vector.

4 Ciclos `for`

for-end

for-end



4 Ciclos `while`:

4 Ciclos `while`:

4 Ciclos `while`:


```
while expresión
    instrucciones
end
```

4 Ciclos `while`:

`while` *expresión*
    instrucciones
`end`

ejecuta instrucciones repetidamente hasta que la parte real de todas las entradas de las (matriz evaluada) expresiones son no-cero.

while-end

while-end

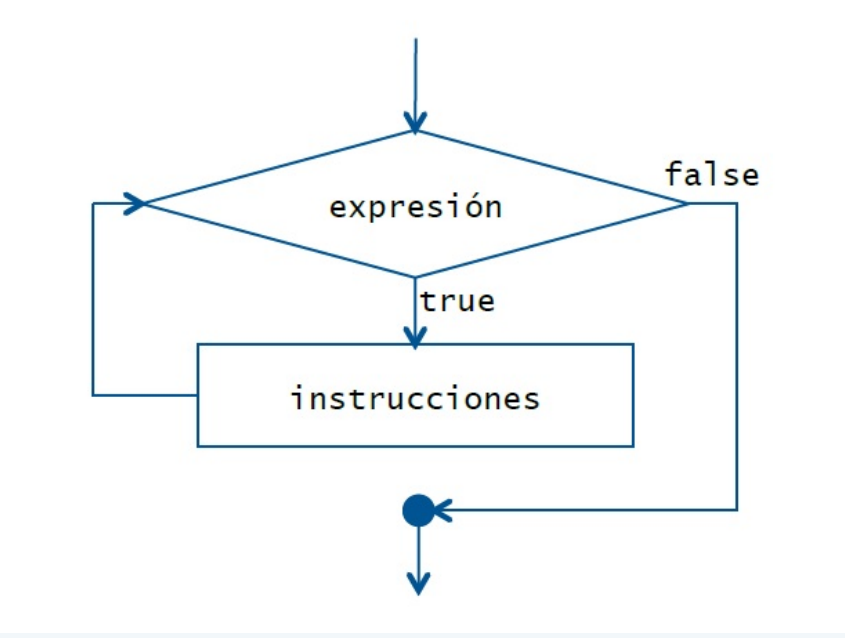


Figura: Diagrama de flujo de la repetición condicional.

4 Interrupción de ciclos/procesos:

4 Interrupción de ciclos/procesos:

4 Interrupción de ciclos/procesos:

- 'break' detiene la ejecución de un `for` o de un `while`.

4 Interrupción de ciclos/procesos:

- 'break' detiene la ejecución de un `for` o de un `while`.
- 'continue' salta a la instancia siguiente de un `for` o de un `while`.

4 Interrupción de ciclos/procesos:

- 'break' detiene la ejecución de un `for` o de un `while`.
- 'continue' salta a la instancia siguiente de un `for` o de un `while`.
- 'return' deja la función en curso para retornar control a la función invocada.

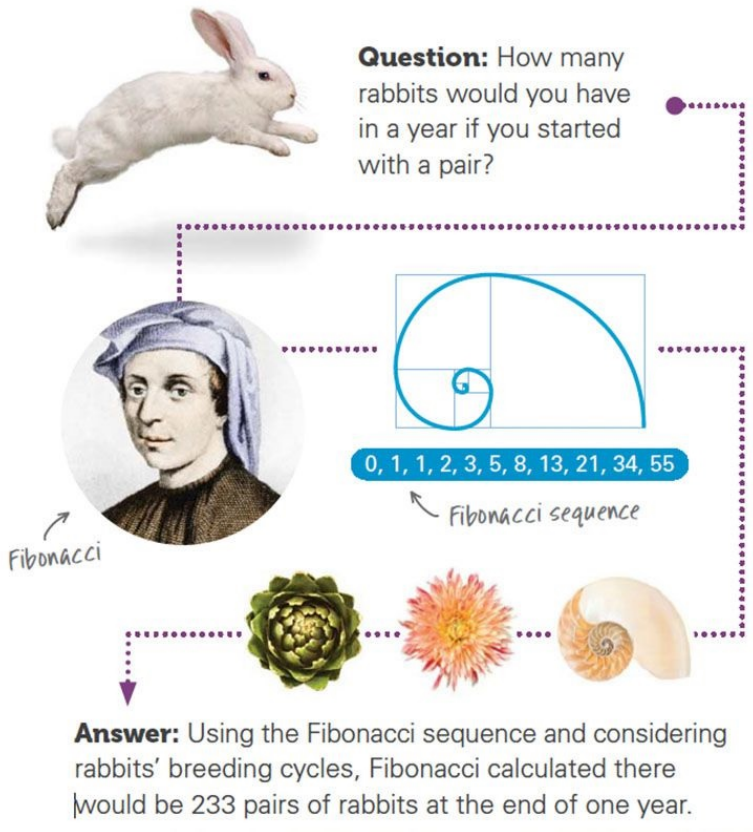
5 Contenidos

Sucesión de Fibonacci

Programe de alguna forma la sucesión de Fibonacci con términos iniciales f_1, f_2 cualesquiera tales que pueda obtener el término $n-$ésimo f_n de dicha sucesión.

Sucesión de Fibonacci

Programe de alguna forma la sucesión de Fibonacci con términos iniciales f_1, f_2 cualesquiera tales que pueda obtener el término $n-$ésimo f_n de dicha sucesión.



Área de un triángulo

Escriba una M-función `function [A] = AreaTri(a, b, c)` que calcule el área de un triángulo a partir de las longitudes de sus lados:

Área de un triángulo

Escriba una M-función `function [A] = AreaTri(a, b, c)` que calcule el área de un triángulo a partir de las longitudes de sus lados:

$$A = \sqrt{p(p-a)(p-b)(p-c)}, \quad \text{con } p = \frac{a+b+c}{2}$$

Área de un triángulo

Escriba una M-función `function [A] = AreaTri(a, b, c)` que calcule el área de un triángulo a partir de las longitudes de sus lados:

$$A = \sqrt{p(p-a)(p-b)(p-c)}, \quad \text{con } p = \frac{a+b+c}{2}$$

La función debe emitir mensaje de error en los casos en que no se pueda calcular el área:

Área de un triángulo

Escriba una M-función `function [A] = AreaTri(a, b, c)` que calcule el área de un triángulo a partir de las longitudes de sus lados:

$$A = \sqrt{p(p-a)(p-b)(p-c)}, \quad \text{con } p = \frac{a+b+c}{2}$$

La función debe emitir mensaje de error en los casos en que no se pueda calcular el área:

Área de un triángulo

Escriba una M-función `function [A] = AreaTri(a, b, c)` que calcule el área de un triángulo a partir de las longitudes de sus lados:

$$A = \sqrt{p(p-a)(p-b)(p-c)}, \quad \text{con } p = \frac{a+b+c}{2}$$

La función debe emitir mensaje de error en los casos en que no se pueda calcular el área:

a) Si alguna de las longitudes recibidas es menor o igual a cero;

Área de un triángulo

Escriba una M-función `function [A] = AreaTri(a, b, c)` que calcule el área de un triángulo a partir de las longitudes de sus lados:

$$A = \sqrt{p(p-a)(p-b)(p-c)}, \quad \text{con } p = \frac{a+b+c}{2}$$

La función debe emitir mensaje de error en los casos en que no se pueda calcular el área:

a) Si alguna de las longitudes recibidas es menor o igual a cero;

b) Si se incumple la desigualdad triangular.

Posición de un punto con respecto a un intervalo

Escriba una M-función `function Donde(x,a,b)` que reciba como argumentos de entrada $x \in \mathbb{R}$ y los extremos de un intervalo $[a, b]$, y escriba en la pantalla un mensaje indicando si $x < a$, $x \in [a, b]$ o $x > b$.

Traza de una matriz

Escriba una M-función `function [t] = Traza(A)` que calcule la traza de una matriz cuadrada **A**. Debe emitir mensaje de error cuando la matriz no es cuadrada.

Traza de una matriz

Escriba una M-función `function [t] = Traza(A)` que calcule la traza de una matriz cuadrada **A**. Debe emitir mensaje de error cuando la matriz no es cuadrada.

- Recuerde que la traza de $A \in \mathbb{R}^{n \times n}$ es igual a

Traza de una matriz

Escriba una M-función `function [t] = Traza(A)` que calcule la traza de una matriz cuadrada **A**. Debe emitir mensaje de error cuando la matriz no es cuadrada.

- Recuerde que la traza de $A \in \mathbb{R}^{n \times n}$ es igual a

$$\text{tr}(A) = a_{11} + a_{22} + \ldots + a_{nn}$$

Ubicando puntos en el plano cartesiano

Escriba una M-función `function [n] = Cuadrante(x,y)` que reciba como argumentos las coordenadas de un punto (x, y) del plano y devuelva un número $(1, 2, 3$ ó $4)$ que indique en qué cuadrante del plano se encuentra.

Ubicando puntos en el plano cartesiano

Escriba una M-función `function [n] = Cuadrante(x,y)` que reciba como argumentos las coordenadas de un punto (x, y) del plano y devuelva un número $(1, 2, 3$ ó $4)$ que indique en qué cuadrante del plano se encuentra.

- El programa debe devolver 0 si el punto se encuentra sobre alguno de los ejes coordenados.

Matriz tridiagonal

Una *matriz tridiagonal* es una matriz cuadrada cuyos elementos son solo distintos de cero en la diagonal principal y las diagonales adyacentes por encima y por debajo de esta.

Matriz tridiagonal

Una *matriz tridiagonal* es una matriz cuadrada cuyos elementos son solo distintos de cero en la diagonal principal y las diagonales adyacentes por encima y por debajo de esta.

- En el caso de `MATLAB` sería que las diagonales `k = -1, 0, 1` no son nulas.

Matriz tridiagonal

Una *matriz tridiagonal* es una matriz cuadrada cuyos elementos son solo distintos de cero en la diagonal principal y las diagonales adyacentes por encima y por debajo de esta.

- En el caso de `MATLAB` sería que las diagonales `k = -1, 0, 1` no son nulas.

Escriba un código para `function [ ] = Tridiag(A)` que devuelva un mensaje donde diga si la matriz **A** es o no tridiagonal.

Derivada de un polinomio

En `MATLAB`, un polinomio se representa mediante un vector que contiene los valores de sus coeficientes: el polinomio

$$p(x) = a_1 x^n + a_2 x^{n-1} + \ldots + a_n x + a_{n+1}$$

se representa como el vector fila $c = [a_1, a_2, \ldots, a_n, a_{n+1}]$.

Derivada de un polinomio

En `MATLAB`, un polinomio se representa mediante un vector que contiene los valores de sus coeficientes: el polinomio

$$p(x) = a_1 x^n + a_2 x^{n-1} + \ldots + a_n x + a_{n+1}$$

se representa como el vector fila $c = [a_1, a_2, \ldots, a_n, a_{n+1}]$.

- Escriba una M-función `function [dc] = derpol(c)` que devuelva el vector `dc` de coeficientes de la derivada del polinomio `c`.

¡Muchas Gracias!