

Diplomado : Herramientas de Programación para Ciencias e Ingeniería

Módulo: MATLAB (Clase 2)

Docente: Juan Sebastián Salcedo Gallo

Universidad Nacional de Colombia Sede Manizales

Contenido

- Operadores lógicos y relacionales
- Comandos de propósito general
- Ciclos
- Ejercicios

Operadores Lógicos y Relacionales

a	b	a and b
0	0	1
0	1	0
1	0	0
1	1	1

Operador lógico: AND

a	b	a or b
0	0	0
0	1	1
1	0	1
1	1	1

Operador lógico: OR

a	not a
0	1
1	0

Operador lógico: NOT

Operadores Lógicos : MATLAB

- El operador lógico **AND** se simboliza en MATLAB como (**a & b**) o (**and(a,b)**).

```
>> a = 1;  
>> b = 1;  
>> and(a,b)  
ans = 1  
>> a & b  
ans = 1  
>> |
```

Operadores Lógicos : MATLAB

- El operador lógico **OR** se simboliza en MATLAB como **(a | b)** o **(or(a,b))**.

```
>> a = 1;  
>> b = 1;  
>> a|b  
ans = 1  
>> or(a,b)  
ans = 1  
>> |
```

Operadores Lógicos : MATLAB

- El operador lógico **NOT** se simboliza en MATLAB como **(~(a))** o **not(a)**.

```
>> a = 1;  
>> not(a)  
ans = 0  
>> ~(a)  
ans = 0  
>> |
```

Operadores Relacionales

Operador	Significado
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
==	Igual que
~=	No igual que

El tipo o clase de la variable **ans** resultante es una variable de tipo ***logical***.

Operadores Relacionales : MATLAB

```
>> x = 10;  
>> y = 5;  
>> x < y  
ans = 0  
>> x <= y  
ans = 0  
>> x > y  
ans = 1  
>> x >= y  
ans = 1  
>> x == y  
ans = 0  
>> x ~= y  
ans = 1  
>> |
```


Operadores Lógicos y Relacionales


Existe una diferencia entre `(=)` y `(==)`, el símbolo `(=)` se refiere a una asignación. Ejemplo: asignaremos a una variable `x` el valor de 2. Para esto escribimos `x = 2`.

El símbolo `(==)` hace una comparación y retorna un **resultado booleano**. Es decir, al escribir `(x == 2)` estamos comparando la variable `x` y cerciorándonos de que esta tome el valor de 2, en este caso, la comparación arrojaría `True (1)`, pero si `x` fuese un número diferente de 2. La comparación arrojaría `False (0)`.

Operadores Lógicos y Relacionales

```
>> x = 2
x = 2
>> x == 2
ans = 1
>> x = 3
x = 3
>> x == 2
ans = 0
>> |
```

Comandos de Propósito General

MATLAB Functions and Commands			
demo	uint16	rand	round
help	uint32	rng	mod 
lookfor	uint64	clock	rem
doc	char	randn	sign
quit	logical	randi	sqrt
exit	true	xor	nthroot
namelengthmax	false	intmin	log
who	class	intmax	log2
whos	format	cast	log10
clear	sin	asin	exp
single	abs	sinh	deg2rad
double	plus	asinh	rad2deg
int8	pi	sind	
int16	i	asind	
int32	j	fix	
int64	inf	floor	
uint8	NaN	ceil	

Funciones Trigonométricas

Función	¿Qué hace?
$\dots(x)$	Función trigonométrica con el ángulo expresado en radianes
$\text{sen}(x)$	seno(radianes)
$\text{cos}(x)$	coseno
$\text{tan}(x)$	tangente
$\text{sec}(x)$	secante
$\text{csc}(x)$	cosecante
$\text{cot}(x)$	cotangente

Funciones Trigonométricas

Función	¿Qué hace?
...d(x)	Función trigonométrica con el ángulo expresado en grados.
sind(x)	seno(grados)
cosd(x)	coseno
tand(x)	tangente
secd(x)	secante
cscd(x)	cosecante
cotd(x)	cotangente

Funciones Trigonómicas

Función	¿Qué hace?
$\dots h(x)$	función trigonométrica hiperbólica con el ángulo expresado en radianes
$\sinh(x)$	seno hiperbólico(radianes)
$a\dots(x)$	inversa de la función con el ángulo en radianes
$a\dots d(x)$	inversa de la función trigonométrica con el resultado expresado en grados
$a\dots h(x)$	inversa de la función trigonométrica hiperbólica con el ángulo y el resultado expresado en radianes

Ciclos

Un bucle o ciclo, en programación, es una secuencia que ejecuta repetidas veces un trozo de código, hasta que la condición asignada a dicho bucle deja de cumplirse.

Los principales son:

- Ciclo for
- Ciclo while
- Ciclo do while
- Condicional if , if-else, elseif (si -> entonces, si no -> entonces)
- Condicional Switch (según)

Ciclo for

```
para  $i \leftarrow x$  hasta  $n$  a incrementos de  $s$  hacer  
    instrucciones  
fin para
```

Elementos:

- Variable de control (i)
- Inicialización de la variable de control (Valor inicial)
- Condición de control (Valor final)
- Incremento (Paso de incremento)
- Cuerpo (Código a ejecutarse hasta que se cumpla la condición)

Ejemplo Ciclo for

```
>> %Imprima los numeros del 1 al 10 de 1 en 1
```

```
>> for i=1:10;
```

```
    disp(i)
```

```
endfor;
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

```
8
```

```
9
```

```
10
```

```
>> |
```

```
>> %Imprima los numeros del 2 al 10 de 2 en 2
```

```
>> for i=2:2:10;
```

```
    disp(i)
```

```
endfor;
```

```
2
```

```
4
```

```
6
```

```
8
```

```
10
```

```
>> |
```

Ciclo while

```
mientras condición hacer  
    instrucciones  
fin mientras
```

Se diferencia del ciclo do while en que el ciclo do while se ejecuta al menos una vez, sin importar si se cumple la condición en la primera iteración, mientras que el ciclo while solo se ejecuta si se cumple la condición.

La condición corresponde a una expresión **booleana**. El cuerpo se ejecutará hasta que la condición deje de ser **true (1)**

Ejemplo Ciclo while

```
>> a = 5;  
>> while (a<=5);  
    disp(a);  
    a = a + 1;  
endwhile;
```

5

```
>> a = 0;  
>> while (a<=5);  
    disp(a);  
    a = a + 1;  
endwhile;
```

0

1

2

3

4

5

```
>> a = 0;  
>> while (a<5);  
    disp(a);  
    a = a + 1;  
endwhile;
```

0

1

2

3

4

Declaración if

En programación, una sentencia condicional es una instrucción o grupo de instrucciones que se pueden ejecutar o no en función del valor de una condición

Si (condición) {

Ejecutar cuerpo

}

Ejemplo Declaración if

```
>> Juan = 22;  
>> if Juan == 22;  
    disp("Juan tiene 22")  
endif  
Juan tiene 22
```

```
>> Juan = 40;  
>> if Juan == 22;  
    disp("Juan tiene 22")  
endif  
>> |
```

Declaración if-else

Si (condición){

Ejecutar cuerpo;

}

Si no {

Ejecutar cuerpo;

}

Ejemplo Declaración if-else

```
>> a = 50;  
>> if a == 1;  
    disp("a es igual a 1")  
else;  
    disp("a es diferente de 1")  
endif;  
a es diferente de 1  
>>
```

Declaración else-if

```
if (Condición 1) {  
    Ejecutar cuerpo;  
}  
  
Elseif (Condición 2) {  
    Ejecutar cuerpo;  
}  
  
Else {  
}
```


Ejemplo declaración else-if

```
>> a = 1;
>> if (a==0);
    disp("a es igual a cero")
elseif (a == 1);
    disp("a es igual a 1")
else;
    disp("a no es igual ni a 1 ni a 0")
endif;
a es igual a 1
>> |
```

Declaración Switch

Una sentencia SWITCH transfiere el control a un conjunto de una o más sentencias de asignación de reglas, dependiendo del valor de una expresión. (Se usa mucho en videojuegos, por ejemplo)

```
switch( variable ){  
    case valor1: accion1; (*)  
    case valor2: accion2; (*++)  
    case valor3: accion3; (*)  
    ...  
    case valorN: accionN; (*)  
  
    default: accionD; (**)  
}
```

Ejemplo Declaración Switch

```
grade.m x
1  nota = 3.0;
2  switch(nota)
3  case 3.0
4      disp("Tu desempeño fue regular ;debes mejorar!")
5  case 4.0
6      disp(";Bien!")
7  case 5.0
8      disp(";Perfecto!")
9  otherwise
10     disp("nota no válida")
11 end
12
```

```
Command Window
>> grade
Tu desempeño fue regular ;debes mejorar!
>> |
```

Ejercicios

- ¿Cuánto es la suma de los números enteros entre 0 y 10?
R/. 55.
- ¿Cuánto es la suma de los números enteros entre 0 y 100?
R/. 5050

***Nota:** Realice este ejercicio usando un ciclo **while**.

```
1  n = 0;
2  suma = 0;
3  while (n<=100) ;
4      suma = suma + n;
5      n = n + 1;
6  endwhile
7  disp(suma)
```

Ejercicios

- ¿Cuánto es la suma de todos los números pares entre 1 y 100?
R/. 2550
- ¿Cuánto es la suma de todos los números pares entre 1 y 1000?
R/. 250500
- ¿Cuánto es la suma de todos los números pares entre 1 y 10000?
R/. 25005000

```
1 suma = 0;  
2 for i=1:10;  
3     if (mod(i,2) == 0);  
4         suma = suma + i;  
5     endif  
6 endfor  
7 disp(suma)  
8
```

Ejercicios

- Realice el ejercicio anterior, esta vez usando un ciclo **while**.


```
1 suma = 0;
2 n = 0;
3 while n<=100;
4     if (mod(n,2) == 0);
5         suma = suma + n;
6     endif
7     n = n + 1;
8 endwhile
9 disp(suma)
10
```

Ejercicios

En todo triángulo la suma de la longitud de 2 de sus lados es mayor a la longitud del tercer lado. Cree un programa al cual el usuario pueda ingresar como datos la longitud de cada lado de un triángulo. Determine si este triángulo puede construirse o no e imprima este veredicto en la consola. Es decir, si los valores de los lados son a , b y c , se deben cumplir:

$$a + b > c$$

$$a + c > b$$

$$c + b > a$$

para que el triángulo propuesto por el usuario sea consistente.

```
1 a = 5;
2 b = 6;
3 c = 4;
4 if ((a + b) > c && (a + c) > b && (b + c) > a);
5     disp("se puede construir")
6 else;
7     disp("no se puede construir")
8 endif;
9
```

Ejercicios

- Dada la calificación de un estudiante, dé un veredicto sobre si pasa o no una asignatura. Tenga en cuenta que el sistema de calificación va de 0.0 a 5.0, y que la **nota mínima** para aprobar es 3.0.

```
1  nota = 3.0;  
2  if (nota < 3.0);  
3      disp("El estudiante reprueba la asignatura")  
4  elseif (nota >= 3.0 && nota <= 5.0);  
5      disp("El estudiante aprueba la asignatura")  
6  else;  
7      disp("Nota no válida")  
8  endif  
9
```

¡Nos vemos mañana!