OBJETIVOS DE APRENDIZAJE

- **Estructuras condicionales**
  - **If**
  - **if <condition> ... end**
  - **if <condition> ... else ... end**
  - **Nested If statements (enunciados condicionales anidados)**
  - **if <condition> ... elseif <condition> ... end**
  - **switch <expression> case <exp1> ... case <exp2> ... end**
- **Funciones lógicas: isletter, isempty, ischar, isnumeric, etc.**

EJERCICIO 6.1 PRIMEROS PASOS

Define las variables *X*, *Y* y *Z* (asignándoles, por ejemplo, el valor “0”) y úsalas para expresar las siguientes condiciones en Matlab:

1. *X* es menor o igual que 5, y *Y* es mayor que 25.
2. *X* es igual a 6 o mayor que *Y*.
3. *X* es un número par y menor que *Y*.
4. *X* es mayor que *Y* y que *Z*.
5. *X* está en el intervalo *[4,100]*.
6. *X* es igual a 2, 3, o 4.
7. *X* es igual a ‘b’ o ‘B’.

Prueba todas las expresiones introduciendo diferentes valores en la variables para comprobar que la expresión es correcta.

EJERCICIO 6.2 CALIFICACIONES

El objetivo de este ejercicio es obtener la calificación asociada a una nota numérica. En primer lugar formularemos una versión simplificada del problema. En lugar de la calificación, escribiremos una función que nos devuelva si la nota proporcionada es un aprobado o un suspenso. Para ello es necesario usar la estructura condicional *if ... else ... end*.

```
function qual = getGrade(grade)
% This function determines whether the grade corresponds to a
% PASS or a FAIL qualification
% Format of the call: getGrade(grade)
```

Ejemplo de ejecución en la Venta de Comandos:

```
>>grade = input('Input a correct numeric note [0,10]: ');
Input a correct numeric note [0,10]: 5.5
>>qual = getGrade(grade)
qual = PASS
```

A continuación, implementaremos dos funciones que nos devuelvan la calificación asociada a una nota numérica: *A (70-100%)*, *B (60-69%)*, *C (50-59%)*, *D (40-49%)* or *N/A (below 40%)*. Cada una de estas dos funciones usarán una estructura condicional diferente. En primer lugar, usando condicionales anidados, y, en segundo lugar, usando la estructura *if ... elseif ... end*.

```
function rdo1 = getGradeNestedIf(grade)
% This function returns the qualification for the grade using
% nested if statements
% Format of the call: getGradeNestedIf(grade)


function rdo2 = getGradeIfElseIf(grade)
% This function returns the qualification for the grade using
% if … elseif … end
% Format of the call: getGradeIfElseIf(grade)
```

Ejemplo de ejecución #1:

```
>>grade = input('Input a correct numeric note [0,10]: ');
Input a correct numeric note [0,10]: 6.37
>>qual = getGradeNestedIf(grade)
```

```
qual = B
```

Ejemplo de ejecución #2:

```
>>grade = input('Input a correct numeric note [0,10]: ');
Input a correct numeric note [0,10]: 7.55
>>qual = getGradeIfElseIf(grade)
qual= A
```

Responde a la siguiente pregunta: por qué no es correcto implementar esta función usando un enunciado *switch*?

EJERCICIO 6.3 CONTROL DE ERRORES (ENTRADA DE DATOS)

En este ejercicio el objetivo es detectar cuando la entrada de datos no es correcta. Para ello, la función controlInputError(grade) recibe una nota numérica como input y devolverá un código de error:

```
function error = controlInputError(grade)
% Check whether or not the grade is valid. It returns a
% message describing the error if the grade is not valid.
% Otherwise, the function returns an empty string
% Format of the call: controlInputError(grade)
```

Esta función comprobará si la entrada de datos es correcta usando funciones lógicas (isempty, isnumeric, etc.) y estructuras *if* anidadas. Si no hay error, la función devolverá un string vacío (*''*).

Para ello las siguientes comprobaciones deberán ser implementadas:

- Si la entrada es vacía, devuelve '*Input must not be empty*'.
- Si la entrada no es número, devuelve '*Input must be numeric*'.
- Si la entrada no es positiva (mayor o igual a 0) y menor que 10, devolvera '*Invalid input. Must be in range [0,10]*'.

Ejemplo de ejecución #1:

```
>>grade = input('Input a correct grade [0,10]: ');
Input a correct grade [0,10]: <INTRO>
>>error = controlInputError(grade):
>>disp(error)
Entry must not be empty.
```

Ejemplo de ejecución #2:

```
>>grade = input('Input a correct grade [0,10]: ');
Input a correct grade [0,10]: 'aaa'
>>error = controlInputError(grade):
>>disp(error)
Input must be numeric.
```

Ejemplo de ejecución #3:

```
>>grade = input('Input a correct grade [0,10]: ');
Input a correct grade [0,10]: 11.25
>>error = controlInputError(grade):
>>disp(error)
Invalid input, must be in the range [0,10].
```

EJERCICIO 6.4 CONTROL DE ERRORES (FICHEROS)

En lugar de intentar cargar un fichero directamente, suele ser buena práctica comprobar si el fichero existe en primer lugar, y, si no existe, proporcionar un mensaje de error adecuado para que la persona usuaria pueda saber cuál es exactamente el problema. En este ejercicio escribiremos la siguiente función:

```
function exists = fileExists(fileName)
% Check whether or not the file exists. If it exists, it will return true
% , and, otherwise, it will return false.
```

Prueba la función intentando cargar ficheros que están en tu carpeta actual y con ficheros que no lo están.

Nota: true y false no son strings, sino valores booleanos, por lo que será necesario usar comillas dobles (“) o simples (‘).

EJERCICIO 6.5 LETRA DEL DNI

Se trata de introducir el número del DNI y calcular la letra que le corresponde. Se define una función para validar que el número no esté vacío y que sea numérico. En caso de que se produzca alguno de estos errores, se visualiza un mensaje de aviso. Pero en caso contrario, se obtendrá la letra del DNI, aplicando los siguientes pasos:

a) Dividir el número del DNI entre *23*.
b) Obtener el resto de la división entera (módulo) (el resultado será un número entre *0* y *22*). Añade *1* al resultado obtenido.
c) Dependiendo del resultado, la función devolverá una de la siguientes letras:

Número	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Letra	T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E

El prototipo de la función es el siguiente:

```
function letter = getLetterDNI(number)
% If the number is invalid,
% it returns a message describing the error.
% Otherwise, the message is 'OK' and
% returns the letter of the DNI number.
```

Para ello usa el siguiente vector:

```
letterVector =['T','R','W','A','G','M','Y','F','P','D','X','B'
        ,'N','J','Z','S','Q','V','H','L','C','K','E'];
```

Verifica la correcta implementación de la función usando los siguientes valores: '*A*', '*Q*', '*x*', '*3*', *23*.