# 4. Variables de texto (strings y chars)

- Variables de texto de clase “cadena” (*strings*)
- Variables de texto de clase “carácter” (*char*)
- Funciones más habituales para operar con variables de texto:

## Variables de texto de clase “cadena” (*strings*)

Matlab es también capaz de tratar con variables de texto. Por ejemplo, la siguiente sintaxis guarda la frase *“Estás usando el programa Matlab”* dentro de la variable `txt`:

⊕

```
>> txt = "Estás usando el programa Matlab"
```

```
txt =

    "Estás usando el programa Matlab"
```

Nótese que para definir la variable `txt` hemos escrito el texto entre **comillas dobles.** Cuando se usa esta clase de comillas, la variable que se crea es de clase *“string”* (cadena):

⊕

```
>> class(txt)
```

```
ans= string
```

Las variables de texto de tipo cadena (*string*) se pueden organizar en vectores o matrices igual que si fuesen números. Por ejemplo:

⊕

```
>> palabras = ["esto", "es", "un", "vector","de","texto"]
```

```
palabras =

  1×6 string array

    "esto"    "es"    "un"    "vector"    "de"    "texto"
```

Podemos seleccionar elementos de este vector de la manera habitual en matlab:

⊕

```
>> palabras(1)
```

```
ans=
  "esto"
```

```
>> palabras(4)
```

```
ans=
  "vector"
```

```
>> palabras(2:4)
```

```
ans =

  1×3 string array

    "es"    "un"    "vector"
```

La variable MP que definimos a continuación es una **matriz de texto** de clase *string* (pues utilizamos dobles comillas):

⊕

```
>> MP = ["Esto" "es"; "una" "matriz"; "de" "palabras"]
```

```
MP =

  3×2 string array

    "Esto"    "es"
```

```
    "una"     "matriz"
    "de"      "palabras"
```

```
>> MP(2,:)
```

```
ans =

  1×2 string array

    "una"    "matriz"
```

```
>> MP(3,2)
```

```
ans =

    "palabras"
```

Se puede utilizar el operador suma para combinar varias cadenas en una sola:

```
>> "programar " + "es " + "fácil"
```

```
ans =

    "programar es fácil"
```

Obsérvese que para que queden espacios entre las palabras ha habido que dejarlos explícitamente entre comillas.

## Variables de texto de clase “carácter” (*char*)

Si queremos acceder individualmente a los caracteres que componen una variable de clase *string*, primero hemos de convertir dicha variable a clase *char* (carácter). Por ejemplo, para convertir nuestra variable inicial `txt` a carácter, basta con emplear la función `char()`:

⊕

```
>> txt = "Estás usando el programa Matlab";
>> t = char(txt)
```

```
t =

    'Estás usando el programa Matlab'
```

En principio lo único que parece haber sucedido es que la variable `t` es igual que la variable `txt` salvo que ahora tiene comillas simples en vez de dobles; ello indica que `t` es de clase `char` como podemos comprobar fácilmente:

```
>> class(t)
```

```
ans =

    'char'
```

Las variables de clase `char` se comportan como si fueran vectores de caracteres (letras, números, espacios u otros símbolos) consecutivos. Para seleccionar parte del texto basta indicar las posiciones a extraer; así, si queremos extraer las letras desde la posición 7 a la 13 del texto guardado en `t` bastaría con ejecutar:

⊕

```
>> t(7:13)
```

```
ans =

    'usando '
```

La función `size` nos da las dimensiones de `t`:

```
>> size(t)
```

```
ans =

     1    31
```

es decir, `t` es una matriz de 1 fila por 31 columnas (que es el número de caracteres que contiene).

Las variables de tipo carácter (`char`) se pueden declarar también directamente usando **comillas**

**simples**:

⊕

```
>> t2 = 'Matlab es un lenguaje curioso'
```

```
t2 =

    'Matlab es un lenguaje curioso'
```

⊕ Para concatenar dos variables de tipo carácter en un vector fila, basta usar corchetes igual que cuando se concatenan vectores numéricos:

```
>> [t ". " t2]
```

```
ans =

    'Estás usando el programa Matlab. Matlab es un lenguaje curioso'
```

Obsérvese que hemos añadido "." para que aparezca la separación entre las dos frases.

⊕ Para concatenar dos variables de tipo carácter en un vector columna, deben tener la misma longitud; ya sabemos que `t` tiene longitud 31; la variable `t2` tiene dimensiones:

```
>> size(t2)
```

```
ans =

    1   29
```

Como esta segunda frase tiene 29 caracteres, podemos añadirle dos espacios, para que tenga también 31 caracteres y así añadirla en una matriz junto con la frase `t`:

```
>> t2 = 'Matlab es un lenguaje curioso  ';
>> TT=[t;t2]
```

```
TT =

  2×31 char array
```

```
    'Estás usando el programa Matlab'
    'Matlab es un lenguaje curioso  '
```

`TT` es ahora una matriz de dimensión 2x31. Si seleccionamos, por ejemplo, las columnas de la 15 a la 20 obtenemos:

```
>> TT(:, 15:20)
```

```
ans =

  2×6 char array

    'l prog'
    'enguaj'
```

# Funciones más habituales para operar con variables de texto:

- `strlength(v)`: muestra el número de caracteres del string `v`:

```
>> strlength("¿Cuantos caracteres tiene esta frase?")
```

```
ans =

    37
```

- `int2str(v)`: convierte valores enteros a texto.

```
>> int2str(123456)
```

```
ans =

    '123456'
```

- `num2str(v,n)`: convierte el valor numérico (o matriz) `v` en texto; si `v` tiene decimales y no se especifica valor de `n`, Matlab por defecto convierte `v` a

texto con 4 decimales como máximo. Si se especifica un valor de n Matlab convierte v en texto con un total de n cifras significativas (incluyendo parte entera y decimal). Si v es un vector (matriz) convierte cada valor numérico del vector (matriz) a texto; también convierte a texto los espacios entre los valores numéricos del vector (matriz).

```
>> num2str(2.13456,2)
```

```
ans = 2.1
```

```
>> num2str(2.13456,4)
```

```
ans = 2.135
```

```
>> num2str(21345.36,2)
```

```
ans = 2.1e+04
```

Podemos extraer los términos del 4 al 8, por ejemplo, de la expresión anterior:

```
>> a=num2str(21345.36);
>> a(4:7)
```

```
ans = 45.3
```

Veamos el resultado de aplicar la función `str2num()` a una matriz numérica:

```
>> a = num2str([1 2 3])
```

```
a = 1  2  3
```

Vemos que a es ahora una matriz de caracteres de dimensión 1$$7

```
>> size(a)
```

```
ans =

   1   7
```

Veamos quienes son los sucesivos términos de a:

```
>> a(1)
```

```
ans = 1
```

```
>> a(2)
```

```
ans =
```

```
>> a(3)
```

```
ans =
```

```
>> a(4)
```

```
ans = 2
```

Nótese que al convertir la matriz `[1 2 3]` en caracteres, Matlab considera que entre cada dos valores hay dos espacios en blanco.

- `str2num(s)`: convierte una cadena de texto en número (esto es, convierte, por ejemplo, la cadena "2" en el número 2, o la cadena "1 2 3" en el vector [1 2 3]; si le pasamos un carácter no numérico, devuelve un vector vacío)

- `str2double(s)`: convierte la cadena de texto `s` en un valor númerico de doble precisión. Al igual que la función anterior, si `s` contiene caracteres no numéricos, esta función devuelve un vector vacío.

- `mat2str(v)`: convierte matrices a caracteres; su funcionamiento es similar a `num2str()`, salvo que también convierte a caracteres los corchetes `[` y `]` que se utilizan para definir la matriz:

```
>> a = mat2str([1 2 3]);
>> a(1)
```

```
ans = [
```

- `ischar(v)`: función que nos indica si `v` es una variable tipo carácter (en cuyo caso `ischar` devuelve un 1) o no (en cuyo caso devuelve un 0).

- `strcmp(a,b)`: compara las cadenas `a` y `b`. Si son iguales devuelve un 1, en caso contrario devuelve un 0. Esta función distingue entre mayúsculas y minúsculas: la misma letra se considera distinta si está en mayúsculas que si está en minúscula.

```
>> a="Hola";
>> b="hola";
>> strcmp(a,b)
```

```
ans = 0
```

- `strcmpi(a,b)`: igual que `strcmp`, pero sin distinguir entre mayúsculas y minúsculas.

```
>> strcmpi(a,b)
```

```
ans = 1
```

- `a==b`: compara las cadenas `a` y `b` carácter a carácter, devolviendo un vector formado por unos y ceros (1 en las posiciones en que `a` y `b` coincidan, 0 cuando no).

```
>> a==b
```

```
ans =

  0  1  1  1
```