

# Cadenas de caracteres

```
#include <string.h>  
#include <stdio.h>  
#include <stdlib.h>
```

# Definición

- Las cadenas de caracteres son simplemente arrays que almacenan variables de tipo `char`. La única característica propia de este tipo de array es que se utiliza el carácter nulo (`'\0'`) para indicar el final de la cadena (esto no es necesario en arrays numéricos), de modo que tendremos que reservar una letra más de la que necesitamos.

- Por ejemplo, para guardar el texto "hola" usaríamos

```
char saludo[5]
```

'h'	'o'	'l'	'a'	'\0'
-----	-----	-----	-----	------

# Declaración

- Para declarar una cadena, podemos usar cualquiera de estas tres opciones:

```
char saludo1[] = "hola";
```

```
char saludo2[] = {'h','o','l','a', '\0'};
```

```
char *saludo3 = "hola";
```

# Comparando cadenas: `strcmp`

- Para comparar dos cadenas alfabéticamente (por ejemplo, para ver si son iguales o para poder ordenarlas), usamos

```
int strcmp (const char *cadena1, const char *cadena2);
```

- Esta función devuelve un número entero, que será:

0	si ambas cadenas son iguales
valor negativo	si cadena1 < cadena2
valor positivo	si cadena1 > cadena2

# Asignando a una cadena el valor de otra: **strcpy**

- Cuando queremos dar a una cadena el valor de otra, usamos:

```
char *strcpy (char *cadenaDestino, const char *cadenaOrigen);
```

# Concatenando dos cadenas: **strcat**

- Para añadir una cadena al final de otra (concatenarla), usamos:

```
char *strcat (char *cadenaDestino, const char *cadenaOrigen);
```

# Extrayendo parte de una cadena: `strtok`

- Para dividir una cadena en segmentos (subcadenas) en base a unos delimitadores (p.ej., una coma o un salto de línea) usamos:

```
char *strtok (char *cadena, const char *delimitadores);
```

**Nota:** cuando la misma cadena es analizada con múltiples llamadas a `strtok`, tras la llamada inicial a la función el primer argumento debe ser `NULL`.

- Ejemplo:

```
char cadena[] = "Hola, mundo.";
const char delimitadores[] = " ,.";

strtok(cadena, delimitadores);
strtok(NULL, delimitadores);
```

Salida:

```
Hola
mundo
```

# Lectura de datos: **sscanf**

- `sscanf` es similar a `scanf`, con la diferencia de que los valores para las variables no se leen desde el teclado, sino desde una cadena de texto:

```
int sscanf (const char *buf, const char *formato, ...);
```

- Ejemplo:

```
int dia, anho;  
char diaSemana[20], mes[20], fecha[100];  
  
strcpy( fecha, "Viernes 31 Enero 2020" );  
sscanf( fecha, "%s %d %s %d", diaSemana, &dia, mes, &anho )
```



# Lectura de una cadena desde fichero: `fgets`

- Para leer una cadena de caracteres desde un fichero y guardarlos en una cadena de caracteres, usamos:

```
char *fgets (char *cadena, int tamaño, FILE *fichero);
```

- **Nota:** La función devuelve una cadena; en caso de error, devuelve `NULL`. Lee hasta encontrar un salto de línea (`\n`) o cuando alcanza la longitud máxima que se ha indicado (*tamaño*). El salto de línea sí se guarda.

# Mostrar el valor de una variable: **snprintf**

- `snprintf` es similar a `printf`, pero escribe su salida en la cadena a la que hace referencia el primer argumento, *cadena*. Además, el segundo argumento, *tamano*, especifica el número máximo de caracteres que `snprintf()` puede escribir en la cadena, incluido el carácter final nulo (`'\0'`).

```
int snprintf(char *cadena, size_t tamano, const char *formato, ...);
```

- Ejemplo:

```
char cadena[50];  
char* c = "programacion";  
  
snprintf(cadena, 4, "%s\n", c);  
printf("Cadena: %s\n", cadena);
```

Salida:

Cadena: pro

# Convertir cadena en valor numérico: **atoi**, **atof**

- `atoi` y `atof` forman parte de la librería estándar de C. Permiten convertir una cadena de caracteres en el valor numérico representado (entero o flotante, resp). La cadena de entrada debe contener un número válido (entero o flotante, resp).

```
#include <stdlib>

int atoi (const char *str);
double atof (const char *str);
```

- Ejemplo:

```
int n1;          double n2;
char* c1 = "50";  char* c2 = "3.1416";

n1 = atoi(c1);
printf("string: %s -> integer: %d\n", c1, n1);
n2 = atof(c2);
printf("string: %s -> double: %f\n", c2, n2);
```

Salida:

```
string: 50 -> integer: 50
string: 3.1416 -> double: 3.141600
```