Entrada y Salida sobre consola

- Entendemos por consola al conjunto: teclado + pantalla

- Debe incluirse la biblioteca estándar de funciones:

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

  Garantiza que tendremos las macros de estas funciones.

  Nota: si vamos a utilizar con frecuencia determinadas funciones propias éstas pueden estar compiladas e incluídas en una biblioteca propia o bien podemos incluirlas mediante una sentencia:

  ```
  #include <mibiblioteca.h>
  ```

- En C la E/S está orientada a caracteres, pero también pueden leerse o escribirse *bytes* (se conoce como E/S a bajo nivel)

E/S carácter a carácter: getchar() y putchar()

- **getchar()**: Lee un carácter de la entrada estándar. Generalmente teclado.

- **putchar()**: Escribe un carácter en la salida estándar. Generalmente en pantalla.

```
main()  /* Cambia tipo de letra */ {
     char ch;
     do {
        ch = getchar();
        if (islower(ch)) putchar (toupper(ch));
        else putchar (tolower(ch));
     } while (ch != '.');
     /* Usa un punto para terminar */
}
```

En el ejemplo han aparecido otras funciones de la biblioteca estándar:

- **islower(ca)**: Devuelve 1 si ca es una minúscula. 0 en caso contrario.

- **isupper(ca)**: Devuelve 1 si ca es una mayúscula. 0 en caso contrario.

- **toupper(ca)**: Convierte el carácter ca a mayúscula.

- **tolower(ca)**: Convierte el carácter ca a minúscula.

E/S línea a línea: gets() y puts()

- **gets()**: Devuelve una cadena terminada con nulo en el array de caracteres que recibe como argumento. Al escribir una cadena y pulsar un retorno de carro (*enter*), **gets()** lee la cadena y a continuación le cambia el retorno de carro por el terminador nulo ('\0').

- **puts()**: Imprime en pantalla una cadena de caracteres, que pueden incluir los códigos que llevan asociados la barra invertida, como '\n'.

```
int getnum() {
   char num[80], n;

   do {
      gets(num);
      if (! numero(num)) {
         puts ("Debe ser un numero\n");
         n = 0;
      }
      else n = 1;
   } while (!n);
   return (atoi(num));
}
```

En el ejemplo anterior **atoi()** convierte una cadena de caracteres ASCII a entero. Otros ejemplos de funciones de la misma familia es **atof()**, ASCII a float.

Entrada y Salida formateada por consola

Salida formateada: printf()

- *printf ("cadena-de-control", lista-de-argumentos);*

La cadena de control contiene tanto caracteres como comandos de formato. De tal forma que los comandos de formato y los argumentos se aparean por orden.

Código de printf()	**Formato**
%c	Un único carácter
%d	entero
%e	Notación científica
%f	Real simple precisión
%g	%e o %f, la que sea más corta
%o	Octal
%s	Cadena de caracteres
%u	Decimal sin signo
%x	Hexadecimal

Los códigos de control de formato pueden tener modificadores que especifiquen la anchura de campo, el número de decimales y un indicador de ajuste a la izquierda.

```
Sentencia printf()            Salida
("%-5.2f", 123.234)          |123.23       |
("%5.2f", 3.234)             |         3.23|
("%10s", "hola")             |         hola|
("%-10s", "hola")            |hola         |
("%5.7s", "123456789")       |      1234567|
```

Entrada formateada: scanf()

- *scanf ("cadena de control", lista de argumentos);*

La cadena de control está formada por códigos de formatos de entrada, que están precedidas por un signo %.

Código de scanf()	Formato
%c	un único carácter
%d	un entero
%e	un número en coma flotante
%f	real simple precisión
%h	un entero corto
%o	un número en octal
%s	una cadena de caracteres
%x	un número en hexadecimal

Los comandos de formato pueden utilizar modificadores de longitud de campo y son números enteros colocados entre el % y el código de comando de formato.

Los datos de entrada tienen que estar separados por espacios en blanco, tabuladores o cambios de línea. Los signos de puntuación como los puntos y comas, los puntos y las comas no sirven como separadores.

Todas las variables utilizadas para recibir valores a través de scanf() deben pasarse por referencia.