Explicación comportamiento y uso de scanf()

- Cuando utilizamos scanf() para leer de la consola, el programa no lee directamente el texto tal cual lo introduce el usuario, sino que éste se almacena en una estructura intermedia que llamaremos **buffer**.

- **Cada vez que el usuario pulsa el retorno de línea (\n), se llena el buffer con la línea introducida (incluyendo el carácter '\n')** y en ese momento el texto introducido es leído por el programa teniendo en cuenta el especificador de formato indicado en la función scanf().

- Vamos a verlo con un ejemplo. Imaginemos que queremos utilizar scanf para leer una variable de tipo carácter, es decir:

  scanf("%c", car);

  E imaginemos que el usuario escribe en la consola "a\n", es decir el carácter 'a' y un retorno de línea ('\n'). Ocurriría que al escribir el retorno de línea ('\n') se rellenará el buffer con toda esa información y se enviará al programa (incluyendo el \n). Nuestro programa, que espera leer un tipo char, se quede simplemente con el carácter 'a' por lo que el carácter '\n' se queda en el buffer pendiente de leer, a la espera de que la próxima instrucción del programa lea algo de la entrada estándar.

- **Por este motivo, si a continuación de un scanf() queremos leer algo más debemos asegurar haber vaciado el buffer de posibles caracteres pendientes de leer (típicamente el \n).**

- **Utilizaremos por ello la función getchar(),** para "consumirlo", es decir para leer ese '\n', pues de lo contrario sería encontrado por el próximo scanf() que se ejecutara, lo que le confundiría y podría alterar el funcionamiento de nuestro programa.

Ejemplo lectura de scanf()

- Supongamos las líneas de código indicadas más abajo y que cuando se le piden los datos al usuario, éste introduce: un número y retorno de línea, después otro número y retorno de línea, un carácter y un retorno de línea y por último otro carácter y un último retorno de línea.

  Por ejemplo: “57\n32\nJ\n”

- La secuencia de lectura sería la siguiente:

Entrada (usuario escribe)	Buffer antes de leer	Instrucción	Dato leído	Buffer después de leer
57\n	57\n	`scanf("%d", &i1);`	57	\n
32\n	\n32\n	`scanf("%d", &i2);`	32	\n
J\n	\nJ\n	`scanf("%c", &c1);`	\n	J\n
	J\n	`scanf("%c", &c1);`	J	\n

Recordad: Cuando se utiliza la consola, el programa no lee directamente el texto tal cual lo introduce el usuario si no que éste se almacena en una tabla intermedia que llamaremos buffer. Cada vez que el usuario pulsa el retorno de carro, se llena el buffer con la línea introducida (incluyendo el carácter '\n') y se envía al programa.

```
int i1, i2;
char c1, c2;

printf("Introduce los datos: \n");
scanf("%d", &i1);
scanf("%d", &i2);
scanf("%c", &c1);
scanf("%c", &c2);

printf("valor de i1: %d\n", i1);
printf("valor de i2: %d\n", i2);
printf("valor de c1: %c\n", c1);
printf("valor de c2: %c\n", c2);
```

Ejemplo de lectura scanf()

- El primer scanf tiene que leer del buffer hasta que encuentra un número (%d).
- En cuanto encuentra el 57 termina de leer, y deja en el buffer un '\n'.
- El segundo scanf, tras la entrada del usuario, se encuentra con \n32\n, y tiene que realizar la misma tarea que el anterior, leer un número. En este caso scanf() se salta el primer '\n', y lee el número entero 32, dejando de nuevo un carácter '\n' en el buffer.
- El tercer scanf(), y tras haber introducido el usuario los datos: J\n, encuentra en el buffer: \nJ\n, es decir, el primer '\n' que es el que quedó anteriormente sin leer, más lo que acaba de introducir el usuario. En este caso por tanto, el tercer scanf() lee el carácter '\n' y no el carácter 'J' como nosotros hubiéramos pensado.
- Por último, en el buffer sigue habiendo dos caracteres por lo que el último scanf() lee la 'J' y se queda en el buffer el ultimo '\n'.