Introducción.

Para un sistema de seguridad, se precisa desarrollar un programa en lenguaje C almacenado en un fichero denominado **`test_cifrado.c`** que permita la encriptación y desencriptación de una contraseña que se solicita al usuario.

La **contraseña** está formada obligatoriamente por **5 caracteres** que deben ser obligatoriamente letras mayúsculas de la A a la Z del alfabeto inglés (queda descartada la Ñ y otros grafismos). Una vez tecleada la contraseña, se debe obtener la contraseña encriptada, conforme al método de encriptación que a continuación se explicará.

Objetivos a conseguir.

Dado que el destinatario del código fuente del programa es la empresa que mantiene el sistema de seguridad, ésta nos obliga a desarrollarlo según las siguientes pautas:

- Es obligatorio disponer de una función que **valide** cada carácter tecleado; es decir, para nuestro caso, debe asegurar que el usuario ha tecleado un carácter entre la A u la Z.
- El código debe disponer de una función denominada **encoder()** que debe recibir el carácter a codificar (se conoce como texto en claro), y debe devolver el carácter encriptado. La función incorporará los argumentos necesarios para la encriptación o cifrado, según el método de encriptación que se describirá.
- El código debe disponer de una función **decoder(),** que es inversa a la anterior. Esta función debe recibir un carácter encriptado y debe devolver el carácter desencriptado o texto en claro. La función incorporará los argumentos necesarios para la desencriptación o descifrado, según el método de encriptación que se describirá.
- El método de encriptación o desencriptación usa distintos pasos en los que obtienen resultados intermedios (en particular de la primera fase); es obligatorio desarrollar una **función para obtener los resultados intermedios**. Esto permite validar los resultados intermedios, tanto en el caso de encriptación como en el caso de desencriptación.
- Se debe desarrollar **un programa sencillo** que pida cinco caracteres a encriptar, se consiga el conjunto de caracteres encriptado, y finalmente se descifre el conjunto de 5 caracteres codificador para obtener los caracteres originales, como prueba de que el proceso es correcto tanto en el sentido de encriptación como en desencriptación.
- Cada función aportada debe ir acompañada de un **comentario** de función estándar y estructurado, como el que se adjunta en el ejemplo siguiente. Véase resumen del tema 3

```
/*
*****************************************************************************
Función:        encoder()
Objetivo:       realiza la encriptación por el método de dos fases para un carácter aportado
                como argumento.
@author:        fjgmoya@ubu.es (tu email)
@date:          01/03/2024
@version:       1.0
@param:         argumento 1 tipo1 por valor / por referencia qué significa
                argumento 2 tipo2 por valor / por referencia qué significa
@return         valor1 retornado tipo1 qué significa
*****************************************************************************
*/
```

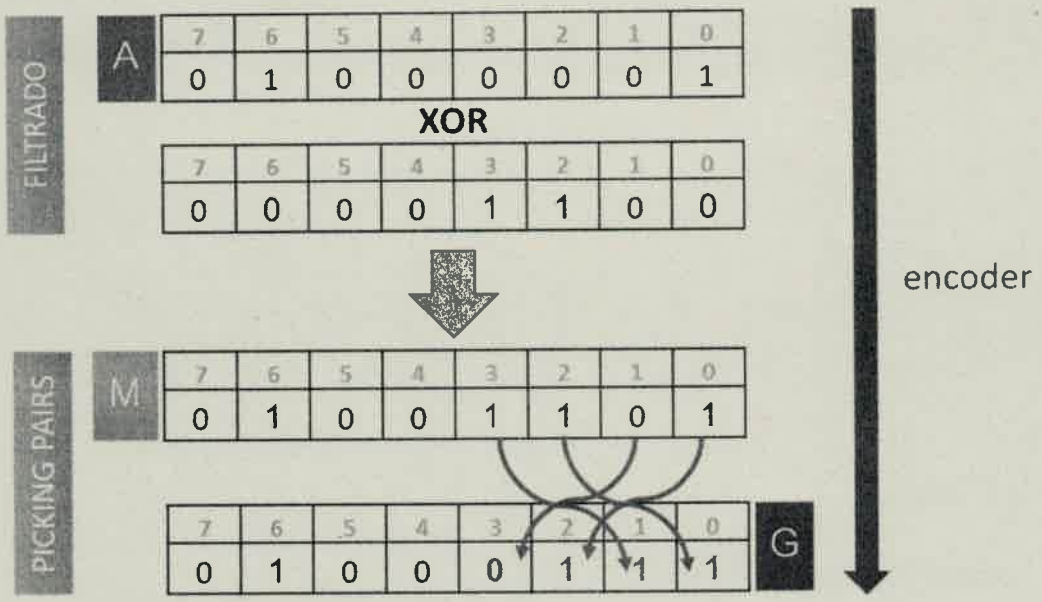
Conocimientos básicos sobre manejo de bits en C.

Operaciones AND, OR, NOT, XOR y desplazamientos a nivel de bit. Ejemplos de uso

```
Int main() {
        char letra, mascara;
        letra='A';
        mascara=5;

        printf("Resultado de AND: %c", letra & mascara); //0100 0001 and
0000 0101 = 0000 0001 => 1 decimal
        printf("Resultado de OR: %c", letra | mascara); //0100 0001 or 0000
0101 = 0100 0100 => 68 decimal
        printf("Resultado de NOT: %c", !letra ); //not(0100 0001)  = 1011
1110 => 180 decimal
        printf("Resultado de XOR: %c", letra ^ mascara); //0100 0001 xor
0000 0101 = 0100 0100 => 68 decimal
        printf("Resultado de desplazamiento hacia la derecha (2 posiciones)
: %c", letra >> 2); //0100 0001 >> 2  = 0001 0000 => 16 decimal
        printf("Resultado de desplazamiento hacia la izquierda (2
posiciones) : %c", letra << 2); //0100 0001 << 2  = 0100 0000 => 64
decimal
        return 0;
}
```

Método de encriptación.



El método de encriptación que se debe implementar utiliza el carácter a codificar y una **máscara**, y se desarrolla en dos fases. La primera se denomina **filtrado**, y la segunda se conoce como **"picking pairs"** o intercambio de bits. La máscara es un valor inalterable en todo el proceso. La máscara que se debe emplear es el valor **12** decimal.

El proceso de **dos fases se explica con detalle a continuación:**

Fase 1. filtrado: consiste en aplicar una operación XOR al carácter aportado con la máscara empleada (12). El resultado del filtrado se conoce como valor filtrado. Por ejemplo:

Carácter 'A' -> 0100 0001 máscara: 12 -> 0000 1100

7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	1

XOR

7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0

=

7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	1

Operación de filtrado: 0100 0001 XOR 0000 1100 = 0100 1101 (77 decimal, valor filtrado correspondiente a la letra 'A'). El ASCII correspondiente al valor 77 conseguido se le conoce como **resultado intermedio del filtrado.**

Fase 2 picking pairs : obtención de las parejas de bits 2-3 y 0-1 e intercambio de las mismas. Si numeramos de izquierda a derecha los bits de cada carácter, podemos identificar rápidamente los bits 2-3 y 0-1:

7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	1

PICKING PAIRS =

7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	1

Una forma sencilla de poder aislar dos bits para después poder intercambiar es mediante una operación de AND con una máscara que aporte valor 1 en los bits que nos interesan. Y a continuación, realizar un desplazamiento hacia la derecha o a la izquierda dos posiciones, para ubicar estos bits en las posiciones 0-1. Por ejemplo, para poder aislar los bits 2-3 procederíamos así, realizando finalmente un desplazamiento hacia la derecha de dos posiciones.

7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	1

AND

7	6	5	4	3	2	1	0
0	0	0	0	**1**	**1**	0	0

=

7	6	5	4	3	2	1	0
0	0	0	0	**1**	**1**	0	0

>> 2

7	6	5	4	3	2	1	0
0	0	0	0	0	0	**1**	**1**

Resultado pp-1

Para el caso del desplazamiento de los bits 0-1 operaríamos de forma similar, pero en este caso cambian la máscara y el desplazamiento, que sería a la izquierda:

7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	1

AND

7	6	5	4	3	2	1	0
0	0	0	0	0	0	**1**	**1**

=

7	6	5	4	3	2	1	0
0	0	0	0	0	0	**0**	**1**

<< 2

7	6	5	4	3	2	1	0
0	0	0	0	**0**	**1**	0	0

Resultado pp-2

Finalmente, una operación de OR entre ambos resultados parciales pp-1 y pp-2 nos ofrece el número con los pares de bits 0-1 y 2-3 intercambiados.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	**1**	**1**

Resultado pp-1

OR

7	6	5	4	3	2	1	0
0	0	0	0	**0**	**1**	0	0

Resultado pp-2

=

7	6	5	4	3	2	1	0
0	0	0	0	**0**	**1**	**1**	**1**

Resultado intermedio de picking pairs

Conseguir el binario original (uniendo los bits 7,6,5 y 4) al resultado del "picking pairs" puede resultar sencillo empleando dos operaciones de desplazamiento y un OR con el resultado anterior:

7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	1

Binario antes de picking pairs

>> 4

7	6	5	4	3	2	1	0
0	0	0	0	**0**	**1**	**0**	**0**

<< 4

7	6	5	4	3	2	1	0
0	**1**	**0**	**0**	0	0	0	0

OR

7	6	5	4	3	2	1	0
0	0	0	0	**0**	**1**	**1**	**1**

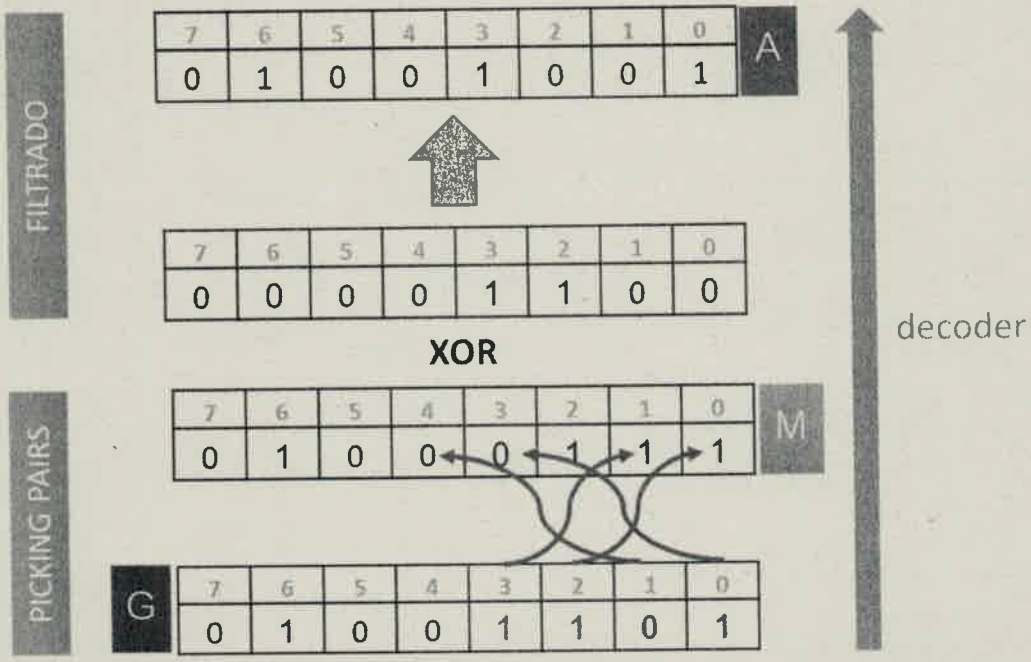
=

7	6	5	4	3	2	1	0
0	**1**	**0**	**0**	**0**	**1**	**1**	**1**

Resultado de picking pairs

El resultado de la fase de picking pairs es el valor binario 0100 0111 que se corresponde con el valor decimal 71. El ASCII correspondiente es el carácter 'G', **que es finalmente el carácter codificado para el carácter inicial 'A'** (lo que se buscaba)

Método de desencriptado.



El proceso de desencriptado sigue exactamente las mismas fases que el proceso de encriptado explicado anteriormente, **pero leído al revés**. Es decir: se comienza con una fase de **picking pairs** como primera fase, y se finalizado con un proceso de **filtrado** con la misma máscara (12) como segunda fase.

Para el ejemplo del caso anterior, partimos del carácter **G** previamente obtenido y aplicamos picking pairs:

7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	1

PICKING PAIRS =

7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	1

Al resultado anterior le aplicamos el XOR de **filtrado:**

7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	1

XOR

7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0

=

7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	1

El binario obtenido 0100 0001 **se corresponde con el ASCII 65, correspondiente al carácter 'A' original.**