Tomás Juan Usón y Sergio Rodríguez Carnicer

1. Introducción:

La práctica ha consistido en la creación de un programa que permite cifrar y descifrar texto plano o cifrado utilizando los cifrados César o Vigenere, a elección del usuario. El programa no soporta caracteres especiales, y los números no se cifran sino que se mantienen con su valor original. El programa soporta espacios en las frases, pero no en la clave, y distingue entre mayúsculas y minúsculas. Además, en el César, debe poder soportar cualquier número de rotaciones, por muy alto o muy bajo que sea, incluyendo números negativos.

Este programa resulta muy útil y fácil de usar para cifrar y descifrar cualquier texto que siga las características previamente explicadas. Tiene una interfaz fácil de entender incluso para un usuario que no tiene conocimientos informáticos.

2.Estructura:

El programa cifra o descifra un texto introducido por el usuario. Una vez completada esta labor, pregunta al usuario si quiere continuar. Así se evita que éste tenga que reiniciar el programa si quiere cifrar/descifrar varios textos, pudiendo repetirlo todas las veces que el usuario quiera.

Primeramente, hemos desarrollado el diagrama de flujo de cada una de las diferentes funciones que el programa utilizaría (cipher_caesar, decipher_caesar, cipher_vigenere y decipher_vigenere). Siguiendo esta primera idea, hemos ido desarrollando en código cada una de las funciones, pero corrigiendo algún error o aspecto importante olvidado a la hora de hacer el diagrama de flujo.

Una vez hechas las cuatro funciones, hemos declarado todas las variables necesarias en el archivo “cipher.h”, introduciendo también un bucle en el “main.c” a modo de “menú principal” para lograr que el programa pueda repetirse las veces que el usuario lo solicite.

Por último, y como verificación de que todo funcionaba como debía, hemos ejecutado muchas veces el programa utilizando todas las opciones posibles de cifrado/descifrado y haciéndolo tanto con rotaciones positivas como negativas, y con números tanto altos como bajos. Para comprobar que las funciones de cifrado funcionan, hemos enviado a varios compañeros de clase un mensaje encriptado, y con sus propios programas han sido capaces de descifrarlo sin ningún problema. Hemos hecho lo mismo con el descifrado

Cabe recalcar que hemos creado nuestra propia función strlen: “strlenC”, que realiza la misma tarea que la función original “strlen”; contar el número de caracteres de una cadena, pero implementándola nosotros mismos, y utilizándose en vez de la función strlen original.

3. Decisiones de implementación y diagrama de flujo

Lo primero que hicimos fue plasmar las primeras ideas de cada una de las funciones de nuestro programa en diagramas de flujo. Comenzamos declarando las variables que íbamos a utilizar posteriormente, y después nos centramos en ver de qué forma debían los diferentes controles de flujo implementarse entre sí para que todo funcionase correctamente y cifrase o descifrase textos correctamente siguiendo las instrucciones dadas. Cabe destacar que en nuestro se han usado dos tipos de metodología: con ascii y sin ascii.

Como se puede ver, el Caesar no utiliza los ascii explícitamente, sino que deja a la máquina hacerlo. Sin embargo queríamos más variedad, así que en la codificación Vigenere utilizamos variables que almacenarán los ascii correspondientes para usarlos según su la situación a afrontar.
El diagrama de flujo resultante con todas las funciones y casos posibles solventados es el siguiente:

4. Sección de resultados

```
Vigènere cipher(1)
Vigènere decipher(2)
Caesar cipher(3)
Caesar decipher(4)
Exit(5)

1
write the phrase you want to cipher
Hola como estas 13
write the key word to cipher it
key
Ciphering: [H, o, l, a,  , c, o, m, o,  , e, s, t, a, s,  , 1, 3]

ciphered: Rsjk ayqm iqdeq 13

cipher completed

continue?: yes(y) no(n)
y


Select a mode:
Vigènere cipher(1)
Vigènere decipher(2)
Caesar cipher(3)
Caesar decipher(4)
Exit(5)
```

Vigenere cipher

```
Select a mode:
Vigènere cipher(1)
Vigènere decipher(2)
Caesar cipher(3)
Caesar decipher(4)
Exit(5)

2
write the phrase you want to decipher
Rsjk ayqm iqdeq 13
write the key word used to cipher it
key
Ciphering: [R, s, j, k,  , a, y, q, m,  , i, q, d, e, q,  , 1, 3]

deciphered: Hola como estas 13

decipher completed

continue?: yes(y) no(n)
```

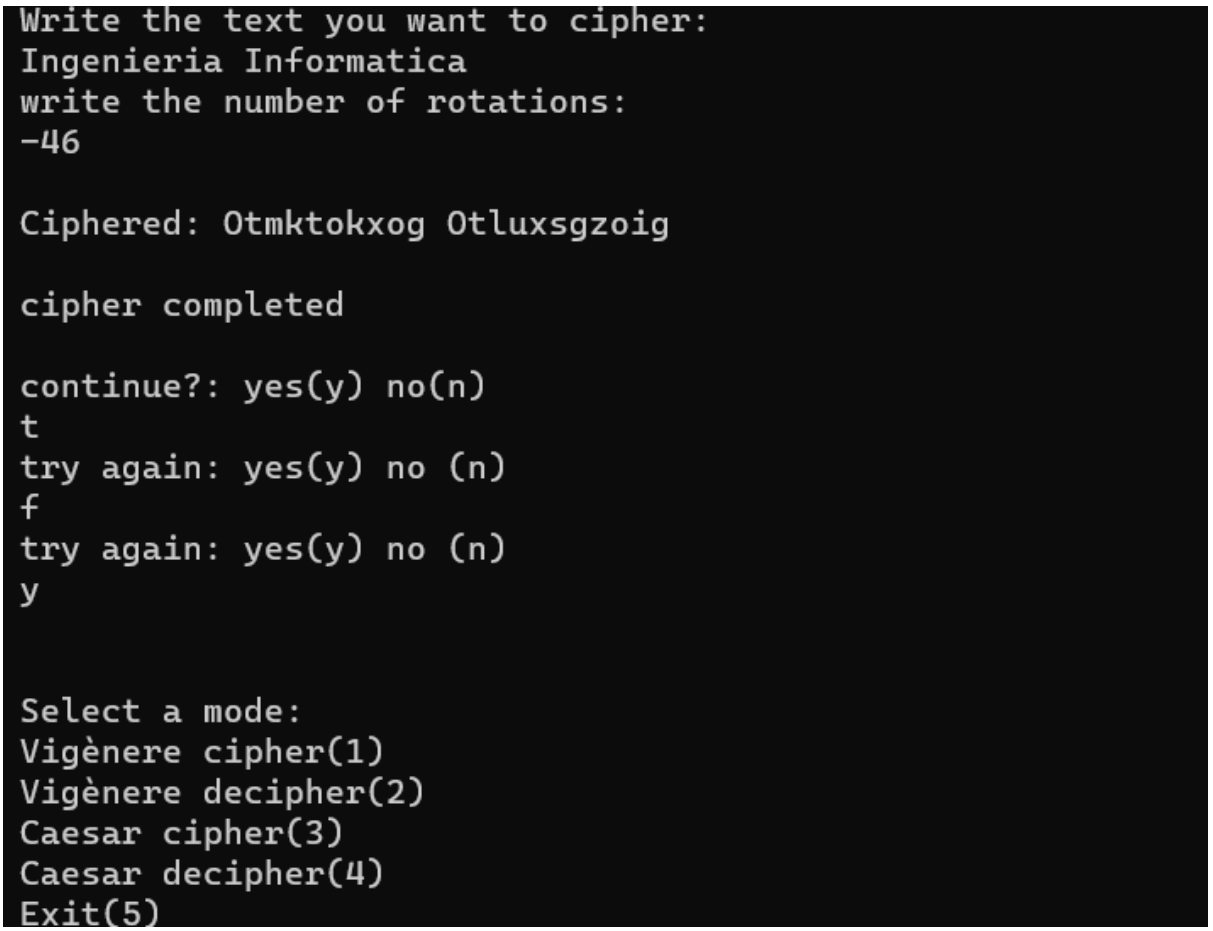
Vigenere decipher

```
Select a mode:
Vigènere cipher(1)
Vigènere decipher(2)
Caesar cipher(3)
Caesar decipher(4)
Exit(5)

3
Write the text you want to cipher:
Hola como estas 13
write the number of rotations:
-4

Ciphered: Dkhw ykik aopwo 13

cipher completed

continue?: yes(y) no(n)
```

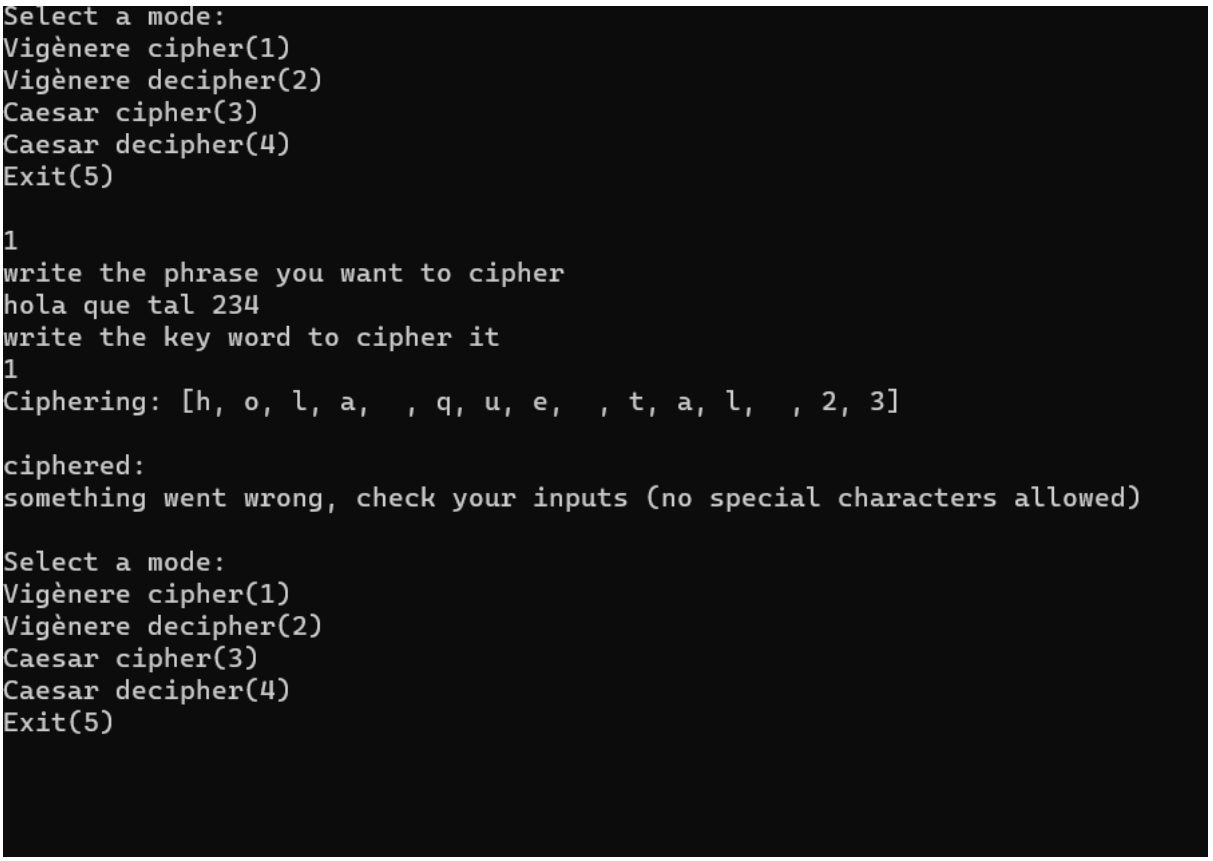
Caesar Cipher

```
Select a mode:
Vigènere cipher(1)
Vigènere decipher(2)
Caesar cipher(3)
Caesar decipher(4)
Exit(5)

4
Write the text you want to cipher:
Dkhw ykik aopwo 13
write the number of rotations:
-4

Deciphered: Hola como estas 13
decipher completed

continue?: yes(y) no(n)
```

Caesar Decipher



Error “try again”



Error “something went wrong, check your inputs”

5. Dificultades, mejoras y conclusión

En este programa hemos encontrado ciertas dificultades a la hora de su realización. Por ejemplo, en el cifrado Vigenere ocurría un error inusual a la hora de dar la vuelta para pasar de la a “a” la “z” si el rango se salía de los límites del Ascii. Esto causaba que los números se volviesen negativos a partir del 128 por algún motivo, pero hemos podido solucionarlo satisfactoriamente con una corrección manual en el código.

En cuanto a las funciones del cifrado César, también hemos tenido que solucionar algunos errores inesperados como saltos de línea espontáneos e inputs que no se guardaban en las variables correspondientes. Para solucionar el tema de los cambios de línea inesperados, se utilizó la función “fflush” para limpiar buffer y evitar estos cambios de línea que dificultan la comprensión de lo que el usuario debía hacer.

Para mejorar el programa, tal vez podríamos buscar la manera de hacerlo más eficiente utilizando menos variables. Al utilizar menos variables el programa funcionará mejor y más rápido.

Este programa es muy útil para comenzar a consolidar conocimientos sobre criptografía y su posible implementación en nuestra futura profesión. En el día a día, la encriptación y

desencriptación de datos está muy presente, por lo que es de vital importancia entender cómo funciona, y tener un mínimo conocimiento sobre su utilización. Por tanto, este programa nos resulta altamente interesante y conveniente tanto para el día a día como para nuestra futura labor profesional.

Cifrado y descifrado Cesar → **Sergio**
Cifrado y descifrado Vigenere → **Tomás**
Función main → **Tomás**
Comentarios Cesar → **Sergio**
Comentarios Vigenere → **Tomás**
Depurado y corrección de código → **Sergio y Tomás**
Traducción → **Tomás**
Ficha técnica → **Sergio y Tomás**