14 de Enero de 2024

MEMORIA TÉCNICA

MEMORIA TÉCNICA DEL PROYECTO FINAL DE FUNDAMENTOS DE LA PROGRAMACIÓN

Realizado por Alejandro Serrano, Alba Prats y Ángela Vinués

ÍNDICE

RESUMEN

El objetivo de esta práctica es crear un gestor de cuentas cifrado. Para este proyecto no ha sido necesaria la realización de diagramas de flujo y pseudocódigo, así que se ha empezado la práctica directamente con el código en lenguaje de programación de "C" utilizando como compilador el programa "CLion".

Para dar comienzo a este proyecto ha sido necesaria la creación de un repositorio en GitHub, y un fichero README el cual consta de actualizaciones constantes de cada modificación del proyecto.

La creación de librerías ha sido fundamental para la correcta estructuración y organización del código. Seguidas de la implementación de diferentes funciones en ellas para modularizar el código y así ahorrar memoria y tiempo de desarrollo.

En conclusión, se ha realizado un algoritmo en C que permite que el usuario pueda almacenar sus cuentas de manera cifrada mediante archivos binarios y encriptación XOR bit a bit, en el que será capaz de, mediante una cuenta maestra, guardar, añadir o eliminar cuentas a su gusto en diferentes ficheros.

1.0 INTRODUCCIÓN

Esta memoria se divide en 4 secciones diferenciadas. En la primera se explicarán las distintas decisiones que hemos tomado en cuanto a la creación e implementación del algoritmo.

En la segunda sección se mostrarán imágenes del programa funcionando, tanto directamente como advirtiendo al usuario de errores como introducir caracteres no válidos.

La tercera brinda especificaciones desarrollo e implementación de la interfaz gráfica, asimismo contiene una breve explicación del proceso de aprendizaje de C#.

En la cuarta, la última, se expone una breve conclusión en la que se explican los errores con los que nos hemos encontrado a lo largo de la realización de la práctica. También se resume el aprendizaje que nos ha brindado superarlos y la importancia de la resiliencia.

2.0 DECISIONES DE IMPLEMENTACIÓN

Para el correcto funcionamiento y compartimentación de nuestro código se ha optado por el uso de dos diferentes librerías, con sus correspondientes archivos .h donde se hayan las cabeceras de sus respectivas funciones y los archivos .c donde se encuentra la implementación de cada función.

- Librería y funciones para la gestión de ficheros

Esta librería alberga dos funciones dirigidas hacia la creación/gestión de ficheros.

- Función writeUserInfo

Esta función tiene como objetivo escribir información introducida por el usuario en un fichero, cifrándola mediante una llamada a una función privada para la librería que realiza el cifrado/descifrado XOR.

En cuanto a sus parámetros, “char* path” es un puntero a una cadena de caracteres que indica la ruta del archivo. “Const userInfo* users” es un puntero a un struct definido en la segunda librería. “Int* numUserPairs” es un puntero a una variable que almacena la cantidad de usuarios y contraseñas que se van a escribir. “Const char* encryptionKey” es un puntero a una cadena con la clave utilizada para cifrar la información.

La función comienza abriendo un archivo binario en modo de escritura (gestionando posibles errores). Tras haber reservado memoria para dos cadenas que contendrán la información antes y después del cifrado, la función las construye incluyendo la cantidad de usuarios, longitudes de nombres de usuario y contraseñas y los propios nombres de usuario y contraseña. La adición del carácter “|” en la cadena cifrada tiene el propósito de actuar como un separador entre las diferentes partes de la información. Finalmente, escribe la cadena cifrada en el archivo y cierra el archivo. La función devuelve 1 en caso de éxito y 0 si hay algún error.

- Función readUserInfo

Esta función tiene como objetivo leer información previamente introducida por el usuario en un fichero binario cifrado, descifrándola mediante una llamada a una función privada para la librería que realiza el cifrado/descifrado XOR.

En cuanto a sus parámetros, “char* path” es un puntero a una cadena de caracteres que indica la ruta del archivo. “Int* numUserPairs” es un puntero a una variable que almacena la cantidad de usuarios y contraseñas que se van a escribir. “Const char* encryptionKey” es

un puntero a una cadena con la clave utilizada para descifrar la información.

La función comienza abriendo un archivo binario en modo de lectura (gestionando posibles errores), en este caso, se ha utilizado un buffer de 1024 bytes para leer el archivo, imprimiéndolo en la consola. Posteriormente, tras descifrar las cadenas y asignarles memoria, se extraen el número de pares de usuarios, nombres y contraseñas y sus respectivas longitudes. Por último, se le asigna memoria al struct y se cierra el fichero.

- Función XORCifrado

Esta función tiene como objetivo implementar un cifrado XOR a una cadena con una clave.

En cuanto a sus parámetros, esta función requiere un puntero a la cadena de datos a cifrar, un puntero a la clave usada, la longitud de ambos parámetros anteriores y un puntero a una cadena donde alojar la información cifrada.

La función consiste en un único bucle donde cada byte de la cadena a cifrar se combina con el byte correspondiente de la clave utilizando la operación XOR (^).

- Función checksum

Esta función calcula el checksum (valor numérico que se calcula a partir de un conjunto de datos con el objetivo de verificar su integridad) de datos introducidos por el usuario.

En cuanto a sus parámetros, esta función requiere de un puntero a una cadena de bytes sobre la que se calculará el checksum y su tamaño.

La función consiste en un único bucle donde se recorre el array de bytes y se calcula el checksum.

- Librería y funciones para la gestión de cuentas

Esta librería alberga un struct donde se agrupan los campos del nombre de usuario, la contraseña, la longitud del usuario y la longitud de la contraseña. También alberga 4 funciones dirigidas hacia la gestión de la memoria dinámica y las cuentas de usuario.

- Función createUserInfo

Esta función asigna dinámicamente memoria para structs y devuelve un puntero a la memoria asignada.

En cuanto a sus parámetros, esta función solo requiere del número inicial de structs a los cuales se les asignará memoria.

- Función fillUserInfo

Esta función tiene como objetivo que el usuario introduzca los datos de una nueva cuenta, almacenándolos en una estructura del tipo "userInfo".

En cuanto a sus parámetros, esta función solo requiere un puntero a un struct "userInfo".

La función comienza pidiendo al usuario que introduzca un nombre de usuario y contraseña para posteriormente rellenar el struct con la información aportada, reservando memoria para ello.

- Función freeAllUserInfo

Esta función tiene como objetivo liberar toda memoria no usada tras asignarla dinámicamente en funciones anteriores.

En cuanto a sus parámetros, esta función requiere de un puntero a un struct "userInfo" y la cantidad de nombres de usuario y contraseñas creadas.

La función consiste en un único bucle donde se llama a una función auxiliar explicada a continuación:

- *Función freeUserInfo*

Dados los mismos parámetros que la función anterior, esta función consiste en un nuevo bucle donde se libera la memoria no utilizada al introducir el nombre de usuario y la contraseña dentro de un struct.

- Función deleteUser

Esta función tiene como objetivo eliminar un determinado usuario, ocupando el espacio en caso de quedar libre con el resto de los elementos del array.

En cuanto a sus parámetros, esta función requiere de un puntero a un struct "userInfo" y la cantidad de nombres de usuario y contraseñas creadas, además del índice que el usuario quiere borrar.

La función comienza verificando si el índice introducido por el usuario es un índice válido, para después llamar a la función freeUserInfo para liberar la memoria del usuario ocupando el índice a borrar. Tras reorganizar el array y restarle 1 al número de nombres de usuario, se ajusta la memoria asignada al array al nuevo número de usuarios.

- Programa principal

En cuanto al código principal, hemos decidido compartizarlo en funciones según el desarrollo del programa para organizarlo.

- Función menuIntro

Esta función es lo primero con lo que se encuentra un usuario. Introduce el programa, da la bienvenida al usuario y le pide que introduzca el nombre de usuario y contraseña maestros.

- Función setupFilePath

Esta función construye la variable del nombre de la ruta del archivo.

- Función welcomeNewUser

Si el usuario es nuevo (no se encuentra un fichero con el nombre de usuario introducido en la función menuIntro), se llama a esta función, donde se le introduce al programa.

Tras el uso de esta función, el programa le volverá a pedir al usuario que introduzca su contraseña, verificando que sean iguales.

- Función menuMain

Esta función presenta las cuatro opciones que el usuario puede elegir para interactuar con el programa.

Tras el uso de esta función y preguntarle al usuario qué opción quiere (comprobando su validez), se seguirá un control de flujo según el número introducido entre 1 y 4.

3.0 RESULTADOS

- Nuevo usuario introduce bien la contraseña maestra

```
||             Encrypted Account Manager Coquette
||                      Welcome!!!                              ||
||     V1.0 | Copyright @ Alejandro Serrano Calvo 2023          ||
||                Date: 2024-01-14 22:11:11                    ||

============================================================
Hi, How should we adress to you?:alba
 Hi, alba
Please, introduce your Master Password:alba

************************************************************
* Welcome! As a new user, let us guide you through the usage *
* of the program:                                           *
*                                                           *
* This program stores user accounts encrypted with the      *
* password you have provided.                               *
*                                                           *
* Once you have completed the initial creation of your      *
* profile, you will be prompted to first enter the username *
* or email of the account to be stored, and then the        *
* password. Once you have finished adding accounts, the     *
* manager will encrypt and save your information.           *
*                                                           *
* As you are creating a new profile, we will ensure that    *
* the entered master password is correct.                   *
************************************************************
Write again your Master Password:alba
 Perfect, your Master Password have been confirmed.
```

- Nuevo usuario introduce mal la contraseña maestra

```
||             Encrypted Account Manager Coquette                 ||
||                    Welcome!!!                            ||
||     V1.0 | Copyright @ Alejandro Serrano Calvo 2023     ||
||                Date: 2024-01-14 22:26:41                 ||

============================================================
Hi, How should we adress to you?: enrique
 Hi,  enrique
Please, introduce your Master Password:24

************************************************************
* Welcome! As a new user, let us guide you through the usage *
* of the program:                                          *
*                                                          *
* This program stores user accounts encrypted with the     *
* password you have provided.                              *
*                                                          *
* Once you have completed the initial creation of your     *
* profile, you will be prompted to first enter the username  *
* or email of the account to be stored, and then the       *
* password. Once you have finished adding accounts, the    *
* manager will encrypt and save your information.          *
*                                                          *
* As you are creating a new profile, we will ensure that   *
* the entered master password is correct.                  *
************************************************************
Write again your Master Password:23
 The password didn't match.Please Introduce a New Master Password:
```

- ~~Añadir una cuenta / Ver una cuenta~~

```
1
What do you want to do?
1) See saved Accounts
2) Add a new Account
3) Delete an Account
4) Save and Exit

Select an Option:
2

Introduce the data of the new Account
Username:@1!ª*/562

Password:funciona?
 What do you want to do?
1) See saved Accounts
2) Add a new Account
3) Delete an Account
4) Save and Exit

Select an Option:
1
Account Number 1:Username: ponnosUn24
Password: porfi
Account Number 2:Username: @1!ª*/562
Password: funciona?
```

- ~~Borrar cuentas~~

```
Select an Option:
3

What account do you want to delete? (1 - 2):
2
What do you want to do?
1) See saved Accounts
2) Add a new Account
3) Delete an Account
4) Save and Exit

Select an Option:
1
Account Number 1:Username: ponnosUn24
Password: porfi
What do you want to do?
1) See saved Accounts
2) Add a new Account
3) Delete an Account
4) Save and Exit
```

- Cifrado en el archivo

4.0 INTERFAZ GRÁFICA

En cuanto a la interfaz gráfica, se intentó realizar con las herramientas que ya conocíamos, utilizando el lenguaje de programación C y el compilador Clion. Para ello, era necesaria la instalación de GTK, más concretamente MSYS2, para poder enlazar las librerías correspondientes (#include <gtk/gtk.h>).

Sin embargo, varias horas de intentar instalarlo sin éxito nos llevaron a valorar otras opciones. Al preguntar a personas de nuestro entorno familiarizadas con interfaces gráficas, se decidió utilizar Visual Studio como entorno de desarrollo, usando C# como lenguaje para evitar la instalación de GTK.

Al ser un lenguaje de programación que nunca se había utilizado anteriormente, la interfaz es muy básica, ya que solo se han aprendido las nociones básicas de C#, centrándose en lo estrictamente necesario para que el programa cumpla sus requisitos. Asimismo, el programa se ha simplificado, puesto que no se ha sido capaz de replicar el código que se tenía en C en C# en su exactitud, por obvia falta de aprendizaje y tiempo, requiriéndose de igual forma de muchísima ayuda de páginas con documentación, personas e inteligencias artificiales para conocer las diferencias entre ambos lenguajes, el funcionamiento de variables que no se habían visto antes y programación orientada a objetos.

- C#, conceptos nuevos para la creación de interfaces gráficas

C y C# son dos lenguajes de programación diferentes, mientras C es un lenguaje de bajo nivel, C# es uno de alto nivel. La principal diferencia enfocándonos en la creación de interfaces gráficas es que C no tiene bibliotecas estándar para el desarrollo de interfaces gráficas. C#, por otra parte, viene con bibliotecas para el desarrollo de interfaces gráficas, como Windows Forms y Windows Presentation Foundation (WPF).

Otra diferencia que se encontró es que C# utiliza un garbage collector que gestiona automáticamente la memoria, liberándola según necesario. Previamente era necesario realizar esta tarea manualmente al acabar una acción.

C#, como se ha mencionado antes, está muy enfocado a programación orientada a objetos, la cual no se ha estudiado todavía, por lo que ha sido un foco de problemas.

En conclusión, al contar con tan poco tiempo y ser un lenguaje que se usa de cero, no se ha podido aprender las nociones básicas tan a profundidad como sería necesario, y se ha saltado directamente a aprender lo estrictamente necesario para crear una interfaz gráfica lo más básica posible, pudiendo llevar esto a no comprender completamente lo que se está haciendo.

- Pasos iniciales para crear la interfaz gráfica

En primer lugar, tras haber descargado Visual Studio se ha abierto un nuevo proyecto en C# (Aplicación de Windows Forms). Lo primero que aparece en la aplicación es un cuadrado donde se añadirán todos los gráficos (botones, etiquetas, cuadros, listas…) arrastrándolos desde el cuadro de Herramientas, Controles comunes.

Abajo a la derecha aparecerán las propiedades, en este caso del cuadrado "Form1". Cuando se arrastre cualquier objeto desde aquí se podrá cambiar el nombre, el texto que aparecerá en él, y otras variables agrupadas en torno al objeto (programación orientada a objetos).

En primer lugar, para empezar con esta interfaz, se añadirán un par de etiquetas donde se escribirá "Introduce su usuario/contraseña", acompañadas de un par de cuadros de texto donde el usuario lo introducirá y un botón que confirme la acción.

Se quiere que determinadas partes del código permanezcan ocultas hasta que se haya realizado alguna acción (se le de clic al botón de confirmar de "Introduce su usuario", por ejemplo), por lo que cambiaremos desde el cuadro de la derecha la propiedad "Visibilidad" de los tres objetos que queremos inicialmente ocultos a "Falso".

Para hacer que cualquier interacción que haga el usuario con los objetos ejecute código, se debe hacer doble clic sobre el objeto que queremos que provoque la acción, viendo entonces la interfaz estándar de cualquier entorno de desarrollo. Por defecto aparece el código básico de los objetos que hemos arrastrado y hecho clic, y se aprecia que las variables tienen tipos como void ya conocidos de programar con C. Además, se aprecia que hay objetos categorizados como public, y otros como private. Public y private determinan desde dónde se puede acceder al miembro en el código, siendo public accesible desde cualquier parte del programa y private solo accesible dentro de la propia clase donde se declara.

Volviendo a la interfaz, al hacer click al objeto que queremos que ejecute código aparecerá el lugar donde introducir ese respectivo código. Aparecen 4 ficheros diferentes, el entorno gráfico que ya se ha visto, "Form1.cs", donde se almacena el código del entorno gráfico, "Form1.Designer" traduce automáticamente a código las acciones que se realizan en el entorno gráfico como arrastrar un botón a una determinada posición, cambiar propiedades, etc. "Program.cs" sería un equivalente al main de Clion, es el lugar donde se declaran variables globales y se aloja el código que lanza en primer lugar el formulario. En cuanto a las variables globales, estas van definidas dentro de una clase.

- Desarrollo del código dentro de la interfaz gráfica

Primero, es importante mencionar que la interfaz gráfica no está completa por errores de falta de conocimiento sobre C#, interfaces gráficas y Visual Studio. Está completo y

funciona el bloque de código de crear el fichero, añadir usuarios, listar usuarios y guardar y salir. No comprueba la contraseña maestra, existen errores en el borrado de cuentas, no calcula el checksum ni cifra.

A continuación, se explica la lógica y decisiones de implementación del código de la interfaz gráfica.

Tras añadir librerías se inicializa el Form y esta espera interacciones. La primera interacción posible es el botón "OK" para introducir el usuario maestro, si no se ha introducido texto en el cuadro de texto, el programa no continúa y aparece una etiqueta que vuelve a preguntar por el usuario. Si por el contrario se ha introducido texto, se vuelven visibles una serie de etiquetas, cuadros de texto y botones que piden la contraseña maestra. Nuevamente, al darle a "OK" tras introducirla se continúa con el programa, volviendo invisibles los objetos usados anteriormente.

Tras comprobar la existencia del fichero, lo abre en modo lectura. La lógica de esta parte del código y de otras varias es muy parecida si no igual a la presentada en C en Clion, traducida a C#. En esta parte del código aparecen varios "MessageBox" y otras cosas que sirven para depurar. Si el fichero no existe, lo crea y añade en la primera línea el usuario y la contraseña maestras, separadas por un carácter "/", además de volver visibles las siguientes partes del programa: listar, añadir y borrar usuarios y salir.

Posteriormente, aparece un intento de cifrado con la misma funcionalidad XOR que no se ha logrado implementar por incapacidad de convertir los tipos de datos para que "%" y "^" funcionen.

Por último, tenemos el desarrollo de la lógica de cada opción disponible. Esta lógica es muy parecida a la que se ha visto en C, añadiendo partes de código propias de C#, "traduciendo" un lenguaje a otro.

"Listar Usuarios" abre un listBox y presenta los usuarios añadidos. El programa lee línea a línea, y deshecha a partir del carácter separador "/" para que no aparezcan las contraseñas.

"Añadir Usuario" lee el fichero, añade la información introducida por el usuario en objetos ya utilizados previamente (cambiando las etiquetas) y cierra el fichero. El programa, al leer información la almacena en una cadena de strings llamado "líneas". Al escribirlo en el fichero es vital añadir un salto de línea para que se almacene un solo usuario/contraseña por línea.

"Borrar Usuario" permite seleccionar un usuario del listBox y lo borra. Sin embargo, añade un usuario más al final del fichero. Este bloque de código se nos ha dificultado más, y no se ha podido corregir el error por falta de conocimientos.

"Guardar y salir" sale del programa.

5.0 CONCLUSIONES

Al ser este proyecto el trabajo de fin de asignatura ha sido notablemente más complicado, por lo que nos hemos encontrado con más retos que superar a lo largo de su realización.

A pesar de que el comienzo no fue difícil, después de implementar el código con las primeras instrucciones para el usuario nos vimos frente al primer error: era necesario implementar algún tipo de separador para que la mayoría de las funciones funcionaran.

Haciendo omisión de todos los errores que hemos tenido que arreglar en la realización de este proyecto debido a que, al ser más extenso y complicado, son más cuantiosos, solo se van a añadir los más relevantes que siguieron a este primero.

Tuvimos numerosos errores al implementar la función de freememory, al igual que varios errores de lógica que se solucionaron yendo paso por paso depurando el código y analizándolo.

El último reto al que nos enfrentamos fue añadir la interfaz gráfica, ya que CLion no se enlazaba con la aplicación GTK. Este paso se nos hizo bastante más complicado porque no encontrábamos la manera de arreglarlo, así que decidimos hacerlo en VisualStudio y C#.

En conclusión, esta práctica no solo nos ha servido para superarnos una vez más en conocimientos en respecto a esta asignatura, sino que también nos será útil en nuestro día a día debido al tema principal del proyecto. También nos ha hecho entender mejor los temas previamente vistos en clase dado que nos hemos enfrentado a un reto más complicado y que ha necesitado más investigación para resolverlo.