Práctica II.- Encriptando y desencriptando ficheros.

En esta segunda práctica, vamos a construir un programa que nos permita encriptar y desencriptar cualquier tipo de fichero, utilizando como método de encriptación el "cifrado de César", también conocido como "cifrado por desplazamiento".

El "cifrado de César", es una de las técnicas de encriptación más simples y más utilizadas, la cual debe su nombre a Julio César, quién lo utilizó para proteger sus mensajes militares más relevantes.

Este método de cifrado, se basa en el desplazamiento de los caracteres dentro de un alfabeto dado, en un número de posiciones fijas. En el caso de Julio César, realizaba siempre un desplazamiento de 3 posiciones sobre cada letra que quería transcribir. Utilizando el método de Julio César sobre el alfabeto castellano, la letra "A" pasa a ser la letra "D", la "B" pasa a ser la "E", la "C" se convierte en una "F"…. y la "Z" pasa a ser la "C". (Al terminarse el alfabeto, se comienza de nuevo y de forma circular por la primera letra del mismo.)

Este método de encriptación entra dentro de los denominados "de clave simétrica", ya que utilizamos la misma clave para encriptar y para desencriptar la información. Nosotros realizaremos un programa que permita no sólo cifrar ficheros de texto, sino cualquier tipo de ficheros, basándonos para la ocultación de la información en el mismo sistema que utilizaba Julio César, pero cambiando el alfabeto y, por supuesto, la clave de encriptación.

Para llevar a cabo la encriptación de nuestros ficheros, utilizaremos como alfabeto base los 256 caracteres que componen el código ASCII extendido. De esta forma, nuestro programa deberá leer cada fichero como si se tratase de un conjunto de caracteres ascii para, posteriormente, proceder a su encriptación/desencriptación. En cuanto a la clave de encriptación/desencriptación, será diferente para cada fichero, y deberá ser calculada por vuestro programa de la siguiente forma: la clave para encriptar/desencriptar cada fichero, será igual al número de repeticiones del carácter ascii más repetido dentro del fichero. Supongamos que en un determinado fichero, el carácter ascii más repetido es la letra "b" con 475 repeticiones. Eso quiere decir que, para ese fichero, su clave de encriptación/desencriptación será el valor 475.

Detalle del funcionamiento de nuestra práctica.

Nuestra aplicación **deberá aceptar n ficheros**, siendo n un valor desconocido en tiempo de compilación. Introducida toda la lista de ficheros (que terminará cuando el usuario introduzca un 0), el usuario introducirá las opciones para que nuestro programa sepa lo que debe hacer con cada fichero, 0 para encriptar, 1 para desencriptar y -1 para finalizar de introducir opciones**. (Es obligatorio realizar el control de errores de entrada/salida.)**

Nuestra aplicación deberá pedirnos, **en este orden**, los siguientes datos:

1.- **Nombre de todos los ficheros que vamos a tratar**, separados cada uno de ellos por un "intro". Cuando introduzcamos un "0" (cero), el programa entenderá que hemos terminado de introducir el

nombre de todos los ficheros. (El nombre de cada fichero deberá ser válido según las especificaciones detalladas más adelante.)

2.- A continuación, mostrará el mensaje "Introduzca todas las opciones deseadas separadas por "intro": 0 Encriptar, 1 Desencriptar y -1 para terminar.", y **aceptará una lista de ceros y unos, terminada con un -1**, separados cada uno de ellos por un "intro". Vuestro programa deberá "casar" cada fichero con la opción introducida, según el orden en que han sido introducidos. Si el número de **ficheros válidos** es mayor que el número de opciones introducidas, se quedarán ciertos ficheros sin tratar. Si por el contrario, el número de opciones introducidas es mayor que el número de ficheros válidos, se desecharán dichas opciones.

Los ficheros de entrada y salida, estarán formados por un nombre y una extensión, según las siguientes especificaciones:

- El nombre de cada fichero contendrá un máximo de 8 caracteres (no se cuenta la extensión).
- La extensión de cada fichero contendrá exactamente 3 caracteres.
- Los ficheros de salida deberán crearse **en el mismo directorio** que los ficheros de entrada. Debe aceptar ficheros de cualquier directorio, eso sí, todos los ficheros de entrada estarán en el mismo directorio. Eso es algo que vamos a dar por supuesto y que no es necesario que comprobéis.
- El nombre de salida para un fichero encriptado, será idéntico al nombre de entrada, y su extensión será cambiada por la extensión "enc". (De esta forma, si encriptamos el fichero "prueba.pdf", su resultado será "prueba.enc").
- El nombre de salida de un fichero desencriptado, será idéntico al nombre del fichero encriptado, cambiando la extensión "enc" por "des". (De esta forma, si desencriptamos el fichero "prueba.enc", el nombre del fichero resultado será "prueba.des").
- Para desencriptar cualquier fichero, su extensión deberá ser "enc". No se desencriptará ningún fichero que no tenga exactamente esa extensión.
- Si el nombre y la extensión de un fichero aparecen repetidos, sólo será tenida en cuenta la primera aparición del mismo, y el resto serán desechadas.
- Aquellos ficheros que no cumplan alguna de estas especificaciones, no serán tratados ni deberán aparecer en el informe final del programa.

Una vez que conocemos los ficheros y sabemos lo que se debe hacer con cada uno, vuestro programa deberá calcular la clave de encriptación/desencriptación de cada fichero, que, tal y como está explicado anteriormente, se corresponderá con el número de apariciones del carácter ascii más repetido en el fichero.

Calculada la clave de encriptación/desencriptación, proceder a encriptar/desencritpar con el método César es algo sencillo. Para encriptar, cada carácter ascii del fichero (que tiene asociado un valor en el código ascii comprendido entre 0 y 255, deberá ser "desplazado hacia posiciones crecientes" del alfabeto, tantos lugares como indique la clave de encriptación. Al llegar al final del alfabeto, se volverá a comenzar circularmente desde el carácter ascii 0. Si lo que se pide es desencriptar, cada carácter ascii del

fichero, será desplazado "hacia posiciones decrecientes" del alfabeto tantas posiciones como indique la clave de desencriptación calculada. En caso de que lleguemos a caracteres inferiores al 0, volveremos a comenzar de forma circular en el carácter ascii 255.

Además de obtener como resultado de vuestro programa los ficheros encriptados/desencriptados, deberá generar un informe cuyo nombre **deberá ser "informe.txt".** Será un fichero de texto, **que será guardado en el mismo directorio que los ficheros de entrada,** en el que para cada fichero que hayamos encriptado/desencriptado, aparecerá una línea en el fichero conteniendo la siguiente información:

NombreFicheroOriginal Opcion(0/1) NombreFicheroSalida ClaveEncriptación/Desencriptación

Cada uno de los campos **estará separado por un espacio en blanco, y cada línea por un "intro" (recordad, una línea por cada fichero tratado).** A continuación os muestro un ejemplo de una determinada entrada, y de la correspondiente salida que debe aparecer en el informe:

"Introduzca el nombre de todos los ficheros a tratar…"

primero.txt

segundo.bmp

tercero.enc

cuarto.des

quinto.mdb

0

"Elija la opción deseada: 0 Encriptar, 1 Desencriptar y -1 para terminar."

3

0

0

1

1

0

-1

El programa "casará" la entrada de la siguiente forma: primero.txt con el primer 0 (desecha la opción 3, ya que no es válida), segundo.bmp con el segundo 0, tercero.enc con el primer 1, cuarto.des con el segundo 1 (pero no será procesado puesto que no se puede desencriptar al no tener extensión de encriptado), y por último quinto.mdb con el último 0. El fichero "informe.txt", contendrá una línea por cada fichero tratado (nótese que no aparece cuarto.des, por el motivo antes mencionado). El último número de cada línea se corresponde con la clave de encriptación/desencriptación):

primero.txt 0 primero.enc 987

segundo.bmp 0 segundo.enc 451

tercero.enc 1 tercero.des 76

quinto.mdb 0 quinto.enc 4

Detalles de la solución que debéis presentar.

El alumno deberá presentar 2 soluciones diferentes al problema propuesto, cada una de ellas en **una unit diferente y utilizando diferente estructura de datos para la resolución, a las que llamaréis unitA.pas y unitB.pas.** El algoritmo para resolver el problema puede ser el mismo en ambas unit, pero la estructura de datos utilizada **debe ser diferente.**

De esta forma, debéis presentar dos programas principales diferentes, a los que llamaréis **fuenteA.pas y fuenteB.pas.** La única diferencia entre fuenteA y fuenteB será que cada uno debe utilizar una unit diferente, de tal forma que, fuenteA contendrá la línea de código "uses unitA;", y fuenteB contendrá la línea de código "uses unitB;". En todo lo demás, fuenteA y fuenteB deben tener un código fuente **EXACTAMENTE IGUAL.** Pueden utilizarse más unidades si se quiere, pero, al menos, deben aparecer en lo que subáis a Agora dos ficheros fuente y dos unidades, tal y como se ha especificado.

Los programas fuente de la práctica (FuenteA.pas y FuenteB.pas), deberán contener, al menos, las siguientes operaciones para el TAD desarrollado (que estarán implementadas en ambas units, cada una con la estructura de datos elegida):

```
VAR
    ListadoDeFicheros:TListadoDeFicheros; {El tipo de variable será declarado en cada unit}
    ElementoFichero:TElemento;
BEGIN
….
InicializarListaDeFicheros(ListadoDeFicheros);
InsertarNombreFicheroEnListado(ListadoDeFicheros, NombreFichero);
EliminarFicheroDeListado(ListadoDeFicheros, NombreFichero);
EsListadoVacio(ListadoDeFicheros);
HaySiguienteFichero(ListadoDeFicheros, ElementoFichero);
ObtenerSiguienteFichero(ListadoDeFicheros, ElementoFichero);
PrimerFichero(ListadoDeFicheros);
CalcularFrecuenciaCaracteres(ElementoFichero);
CalcularClaveEncriptacion(ElementoFichero);
……
END.
```

No se dará por apta ninguna práctica cuyos programas principales (FuenteA.pas y FuenteB.pas) tengan el siguiente código fuente:

BEGIN

EjecutarAplicacion(); ***{PRÁCTICA NO APTA}***

END.

Al menos una de las dos soluciones deberá ser resuelta utilizando memoria dinámica. (Para quienes pretendan resolver todo con memoria estática, sólo recordarles que la entrada debe aceptar n ficheros).

Deberéis realizar obligatoriamente un control de errores de los datos de entrada solicitados, de tal forma que cumplan con TODAS Y CADA UNA de las especificaciones anteriormente descritas.

Observaciones.

- Es obligatorio hacer control de errores de entrada/salida. **No se dará apta** ninguna practica que presente fallos en tiempo de ejecución por motivos de trabajo con ficheros.
- Se prohíbe el uso de las funciones: delay, GotoXY, WhereX, WhereY, Textcolor, TextBackground, sound, Nosound, Exit. También se prohíbe usar las unit Graph y crt.
- **No se permite la utilización de ficheros temporales**. El programa deberá realizar todas las operaciones en memoria, y, al finalizar, creará el fichero encriptado/desencriptado, según lo que haya solicitado el usuario.
- El tamaño del fichero original de entrada, y el del fichero resultado una vez encriptado/desencriptado **deberá ser exactamente el mismo.**
- **No se aceptarán prácticas que presenten efectos laterales en su código.**
- Es **Obligatorio** que el código esté **identado y comentado**.
- Para cada función y procedimiento han de figurar los siguientes datos en los comentarios:
  - Descripción del valor de retorno (en caso de funciones).
  - Descripción de los parámetros de la función o procedimiento (si son de entrada o salida y para qué se utilizan).
  - Pequeña descripción de lo que hace la función o procedimiento.

Normas de entrega

- El programa debe implementarse utilizando el compilador Free Pascal versión 2.2.2 (no sirve otra versión) **en plataforma Linux (NO SIRVE OTRA PLATAFORMA).** Para la corrección se compilará el código fuente y posteriormente se ejecutará el programa. Dicha compilación/ejecución se realizará en una máquina con Linux y el compilador Free Pascal. Si el alumno utilizase otro compilador o sistema operativo para desarrollar el programa **es responsabilidad suya** el comprobar que funciona bajo las condiciones mencionadas.

- Se puede realizar la práctica individualmente o en grupos de **dos personas**. En caso de hacerlo en grupo, se entregará una práctica por grupo.

- La entrega de las prácticas ha de seguir las siguientes normas:

  - Hay que entregar código fuente y los ejecutables **para sistemas Linux**. (Los ficheros con el código fuente principal deberán llamarse "fuenteA.pas" y "fuenteB.pas". Las unidades deberán llamarse "unitA.pas" y "unitB.pas".)

  - Deberéis entregar un documento PDF -de no más de una hoja- en el que detalléis de forma muy breve lo siguiente:

    - 1.- Encabezado con nombre, apellidos y dni del autor/autores de la práctica.

    - 2.- Descripción de las estructuras de datos utilizadas en la unitA, describiendo la solución propuesta.

    - 3.- Descripción de las estructuras de datos utilizadas en la unitB, describiendo la solución propuesta.

  - La entrega se realizará a través de Agora, con la cuenta de uno de los alumnos que la presente, **sólo la de uno**. Debéis subir un sólo fichero **comprimido en zip** que contenga los ficheros fuente, los ejecutables para Linux (deberán ser 2, fuenteA y fuenteB) y el documento PDF. El nombre del fichero será:

    Codificador–DniAlumno1letra–DniAlumno2letra.zip

    Para dos alumnos cuyos DNI sean 12345678-Z y 87654321–X, el fichero se llamará:

    Codificador–12345678Z–87654321X.zip

    Fijaos que la letra está pegada al número del documento. **El formato del fichero comprimido debe ser ZIP.**

Forma de corrección.

Los ficheros de carga se encuentran en Agora. Cada colección de ficheros lleva un documento de texto denominado "listado.txt", que es el que debéis utilizar para probar las baterías de prueba. Dentro de cada carpeta "resultado", aparecen los ficheros tal y como deben quedar después de ejecutar el programa, y podéis comprobar las claves de encriptación/desencriptación en el fichero "informes.txt".

NO SERÁN APTAS aquellas prácticas en las que:

- No sean presentadas 2 soluciones con 2 estructuras de datos diferentes.
- Carezcan del documento pdf con la explicación solicitada.
- Carezcan de los ejecutables de Linux, o, por el contrario, incluyan ejecutables para otros sistemas operativos.

- Presenten fallos en más de 3 baterías de prueba.
- Presenten alguno de los errores anteriormente citados en el apartado “Observaciones”.

Las prácticas serán compiladas, entre otras, con las opciones –Cr y –gh. La primera comprueba que no se producen salidas de rango en vuestras prácticas. La segunda comprueba que liberáis toda la memoria que habéis reservado, **algo que es obligatorio si utilizáis memoria dinámica para resolver la práctica.**

Fechas de entrega:

La práctica sólo tiene una fecha de entrega:

- ✓ **31 de marzo de 2010.**