Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Programación 1

Examen práctico. 30 de enero de 2020

Turno 1.º 15:15 a 16:45

- Tiempo máximo para realizar el trabajo de programación propuesto: 90 minutos
- Entrega del trabajo a través de la plataforma Moodle

El ROT13 es un tipo de cifrado César en el que se utiliza únicamente la clave 13, dado que esta convierte al ROT13 en lo que en términos criptográficos se conoce como un sistema de cifrado recíproco o simétrico: para descifrar un texto cifrado con ROT13, basta con volver a cifrarlo aplicando de nuevo ROT13. ROT13 se utiliza en ocasiones en foros de internet como medio para ocultar de miradas accidentales el final de un chiste, la solución a un acertijo, o el destripe del final de una película o una historia[1]. En estos contextos, es habitual que el contenido cifrado aparezca precedido por un carácter específico (normalmente el carácter '>'), que indica que a partir del mismo y hasta el final de la línea el texto se encuentra cifrado:

```
Cuando George R. R. Martin publique las novelas que finalizan
la serie _Canción de hielo y fuego_, en la que se basa la conocida
serie de televisión _Juego de Tronos_, > pbzcebonerzbf fv un fvqb pncnm,
aunque esto suene cruel, >  qr whagne qr ahrib n ybf crefbanwrf ra
> nytúa fvgvb. Pba ha aúzreb >= dhr 2, abf cbqeínzbf
  >qne pba ha pnagb ra ybf qvragrf.

Un exfan anómino.
```

En el ejemplo anterior, la negrita se ha añadido únicamente para facilitar la lectura y comprensión del mismo.

Se pide diseñar un programa que solicite al usuario los nombres de dos ficheros de texto. El primero debe existir y puede contener cualquier tipo de texto. El programa debe copiar el contenido del primer fichero en el segundo, aplicando el cifrado ROT13 a aquellos caracteres de cada línea que se encuentren detrás de un carácter '>'.

Los caracteres de las líneas en las que no aparece el carácter '>' se copian sin aplicar ningún cifrado, al igual que los caracteres que aparecen antes del carácter '>' en las líneas que sí lo contienen. Si en una línea aparece más de una vez el carácter '>', tras la segunda aparición y sucesivas se debe seguir aplicando el cifrado ROT13 hasta el final de la línea en curso, tal y como se haría si se tratase de la primera aparición.

Al ser ejecutado, si no hay ningún problema con la apertura y creación de los ficheros involucrados, el programa informará de que ha podido generar el nuevo fichero correctamente e indicará, además el número de letras que se han cifrado en el proceso, tal y como se muestra en el siguiente ejemplo:

[1] Basado en «ROT13». (2020, 15 de enero). *Wikipedia, The Free Encyclopedia*. Consultado el 28 de enero de 2020 en https://en.wikipedia.org/w/index.php?title=ROT13&oldid=935941917 y «ROT13». (2020, 11 de enero). *Wikipedia, La enciclopedia libre*. Consultado el 28 de enero de 2020 en https://es.wikipedia.org/w/index.php?title=ROT13&oldid=122663691.

```
Escriba el nombre del fichero a leer: martin.txt
Escriba el nombre del fichero a escribir: rot13.txt
El fichero "rot13.txt" se ha generado correctamente.
Se han cifrado un total de 115 letras.
```

En el caso de que el programa no pueda abrir el primer fichero o no pueda crear el segundo, se limitará a escribir el mensaje de error correspondiente:

```
Escriba el nombre del fichero a leer: george-r-r-martin.txt
Escriba el nombre del fichero a escribir: rot13.txt
No se ha podido acceder al fichero "george-r-r-martin.txt".
```

```
Escriba el nombre del fichero a leer: chiste.txt
Escriba el nombre del fichero a escribir: a/b.txt
No ha podido crearse el fichero "a/b.txt".
```

A diferencia de los programas de la práctica 5 y del trabajo obligatorio, el programa no tiene que preocuparse acerca de la ubicación de los ficheros a leer o escribir. Esto quiere decir que no es necesario concatenar rutar relativas a los nombres de los ficheros ni, en general, nada de lo que se realizaba en la función `pedirNombreFichero` del mencionado trabajo. En los ejemplos de ejecución anteriores, los ficheros involucrados deberían estar y serían creados en el directorio de ejecución del programa solicitado (en el caso de CodeLite 13.0.0, en la carpeta «Debug» del proyecto en el que se ha desarrollado el programa).

El desarrollo del programa solicitado en este examen se puede basar en el código personal del trabajo obligatorio, aunque todo el código necesario para compilar correctamente el programa deberá entregarse en un único fichero denominado «`turno1.cpp`» que contenga tanto la función `main` del programa solicitado como otras en las que pueda haberse apoyado.

En la tarea Moodle en la que hay que entregar el trabajo, se han dejado dos ficheros parcialmente cifrados con ROT13 que pueden utilizarse para hacer pruebas. También pueden realizarse pruebas con cualquier otro fichero, ya que, si se cifra dos veces consecutivas con ROT13, el fichero resultante tras el segundo cifrado tiene que ser idéntico al de partida. Esta última propiedad es una condición necesaria para la corrección del programa, pero no suficiente.

Presentación del trabajo y criterios de evaluación

Cada estudiante presentará, antes de hora que se indique al comienzo del examen, a través de la plataforma Moodle[2] el fichero con el código del módulo principal del programa, «`turno1.cpp`». El contenido del fichero vendrá encabezado por un comentario con el nombre y apellidos del estudiante.

En este examen práctico **se valorará especialmente que el programa pueda ser compilado sin errores y que, al ser ejecutado, proporcione los resultados esperados.**

También se valorará la eficiencia de la solución presentada, su diseño, modularidad, la adecuada especificación de cada una de las funciones que integren la solución y la legibilidad del código atendiendo a los criterios de la *Guía de estilo para programar en C++* publicada en la web de la asignatura.

[2] https://moodle2.unizar.es/

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

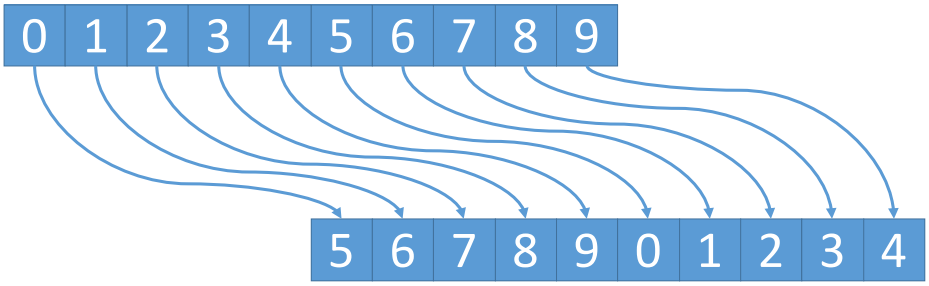
Programación 1

Examen práctico. 30 de enero de 2020

Turno 2.º 17:15 a 18:45

- Tiempo máximo para realizar el trabajo de programación propuesto: 90 minutos
- Entrega del trabajo a través de la plataforma Moodle

El ROT5 es un sistema de cifrado tipo César que se aplica, en lugar de a las letras del alfabeto, a los dígitos del '0' al '9' y en el que se utiliza siempre como clave de cifrado el 5.



La clave con la que se cifra en ROT5 es, precisamente, la que convierte al cifrado César aplicado a dígitos en lo que en términos criptográficos se conoce como un sistema de cifrado recíproco o simétrico: para descifrar un texto cifrado con ROT5, basta volver a cifrarlo aplicando de nuevo ROT5.

ROT5 se puede utilizar para cifrar el contenido numérico de textos, aunque como cualquier técnica de cifrado césar, su seguridad criptográfica es prácticamente nula. En todo caso, puede *disfrazar* ligeramente ciertos números en un texto. Por ejemplo, el número de teléfono de la centralita de la Universidad de Zaragoza, el 976 76 10 00, cuando se cifra con ROT5, no es tan fácilmente reconocible como número telefónico: 421 21 65 55

Se pide diseñar un programa que solicite al usuario los nombres de dos ficheros de texto. El primero debe existir y puede contener cualquier tipo de texto. El programa debe copiar el contenido del primer fichero en el segundo, aplicando el cifrado ROT5 a los dígitos que se encuentren en el mismo. Los caracteres que no sean dígitos, se copian en el segundo fichero tal cual, sin hacer ningún tipo de cifrado ni de modificación.

Al ser ejecutado, si no hay ningún problema con la apertura y creación de los ficheros involucrados, el programa informará de que ha podido generar el nuevo fichero correctamente e indicará, además, el número de líneas del fichero en las que ha cifrado algún carácter:

```
Escriba el nombre del fichero a leer: unizar.txt
Escriba el nombre del fichero a escribir: rot5.txt
El fichero "rot5.txt" se ha generado correctamente.
Se han modificado un total de 4 líneas.
```

En el ejemplo anterior, si el contenido del fichero «`unizar.txt`» fuese el que se muestra en el cuadro de debajo, a la izquierda, el contenido del fichero «`rot5.txt`» generado por el programa debería ser el que se muestra a su derecha:

«unizar.txt»	«rot5.txt»
UNIVERSIDAD DE ZARAGOZA Pedro Cerbuna 12 50009 Zaragoza - España Información: 976 76 10 00 ciu@unizar.es https://unizar.es:443/index.html CIF: Q-5018001-G	UNIVERSIDAD DE ZARAGOZA Pedro Cerbuna 67 05554 Zaragoza - España Información: 421 21 65 55 ciu@unizar.es https://unizar.es:998/index.html CIF: Q-0563556-G

En el caso de que el programa no pueda abrir el primer fichero o no pueda crear el segundo, se limitará a escribir el mensaje de error correspondiente:

```
Escriba el nombre del fichero a leer: george-r-r-martin.txt
Escriba el nombre del fichero a escribir: rot5.txt
No se ha podido acceder al fichero "george-r-r-martin.txt".
```

```
Escriba el nombre del fichero a leer: unizar.txt
Escriba el nombre del fichero a escribir: a/b.txt
No ha podido crearse el fichero "a/b.txt".
```

A diferencia de los programas de la práctica 5 y del trabajo obligatorio, el programa no tiene que preocuparse acerca de la ubicación de los ficheros a leer o escribir. Esto quiere decir que no es necesario concatenar rutar relativas a los nombres de los ficheros ni, en general, nada de lo que se realizaba en la función `pedirNombreFichero` del mencionado trabajo. En los ejemplos de ejecución anteriores, los ficheros involucrados deberían estar y serían creados en el directorio de ejecución del programa solicitado (en el caso de CodeLite 13.0.0, en la carpeta «Debug» del proyecto en el que se ha desarrollado el programa).

El desarrollo del programa solicitado en este examen se puede basar en el código personal del trabajo obligatorio, aunque todo el código necesario para compilar correctamente el programa deberá entregarse en un único fichero denominado «`turno2.cpp`» que contenga tanto la función `main` del programa solicitado como otras en las que pueda haberse apoyado.

En la tarea Moodle en la que hay que entregar el trabajo, se han dejado dos ficheros parcialmente cifrados con ROT5 que pueden utilizarse para hacer pruebas. También pueden realizarse pruebas con cualquier otro fichero, ya que, si se cifra dos veces consecutivas con ROT5, el fichero resultante tras el segundo cifrado tiene que ser idéntico al de partida. Esta última propiedad es una condición necesaria para la corrección del programa, pero no suficiente.

Programación 1

Examen práctico. 30 de enero de 2020

Presentación del trabajo y criterios de evaluación

Cada estudiante presentará, antes de hora que se indique al comienzo del examen y a través de la plataforma Moodle[1], el fichero con el código del módulo principal del programa, «`turno2.cpp`». El contenido del fichero vendrá encabezado por un comentario con el nombre y apellidos del estudiante.

En este examen práctico **se valorará especialmente que el programa pueda ser compilado sin errores y que, al ser ejecutado, proporcione los resultados esperados.**

También se valorará la eficiencia de la solución presentada, su diseño, modularidad, la adecuada especificación de cada una de las funciones que integren la solución y la legibilidad del código atendiendo a los criterios de la *Guía de estilo para programar en C++* publicada en la web de la asignatura

[1] https://moodle2.unizar.es/

Posible solución al problema del turno 1.º

```
/*****************************************************************************\
 * Curso de Programación 1. Examen práctico de enero de 2020
 * Autores: Miguel Ángel Latre
 * Última revisión: 27-2-2020
 * Resumen: Solución al problema planteado en el turno 1
\*****************************************************************************/
#include <iostream>
#include <fstream>
#include <cctype>
using namespace std;

const int MAX_LONG_NOMBRE_FICHERO = 200;
const int MAX_LONG_LINEA = 200;
const int CLAVE_ROT13 = 13;
const char MARCA_CIFRAR = '>';
const int NUM_LETRAS = 'Z' - 'A' + 1;


/*
 * Pre:  desplazamiento >= 0, numCifradas >= 0
 * Post: Si el valor del parámetro «c» es una letra mayúscula o minúscula
 *       del alfabeto inglés, ha devuelto la mayúscula o minúscula del
 *       alfabeto inglés correspondiente a aplicar un cifrado de César a
 *       la letra «c» con un desplazamiento igual al valor del parámetro
 *       «desplazamiento». Si «c» es una letra mayúscula, la letra cifrada
 *       devuelta es también una mayúscula; si «c» es una minúscula, la
 *       letra cifrada devuelta es también una minúscula. En caso de que
 *       «c» no sea una letra del alfabeto inglés, ha devuelto «c» sin
 *       hacer ninguna transformación.
 *       Si ha aplicado el proceso de cifrado al carácter «c», el valor del
 *       parámetro «numCifradas» se ha incrementado en una unidad con respecto
 *       al que tenía cuando se ha comenzado a ejecutar la función. En caso
 *       contrario, lo ha dejado inalterado.
 */
char cifrarCesar(const char c, const int desplazamiento, int& numCifradas) {
    if (isalpha(c)) {
        numCifradas++;
        char primeraLetra = 'A';
        if (islower(c)) {
            primeraLetra = 'a';
        }
```

```
        int posicion = c - primeraLetra;
        int posicionCifrada = (posicion + desplazamiento) % NUM_LETRAS;
        return posicionCifrada + primeraLetra;
    }
    else {
        return c;
    }
}


/*
 * Pre:  «fOriginal» y «fCifrado» están abiertos y clave >= 0.
 * Post: Ha copiado el contenido pendiente de leer de «fOriginal» en
 *       «fCifrado», aplicando un cifrado César con un desplazamiento igual al
 *       valor del parámetro «clave» solo a aquellos caracteres de cada línea
 *       que aparecén detrás del carácter «MARCA_CIFRAR». Ha asignado al
 *       parámetro «numCifradas» el número de letras que se han copiado a
 *       «fOriginal» aplicándoseles el cifrado mencionado.
 */
void cifrarFichero(istream& fOriginal, ostream& fCifrado,
                   const int clave, int& numCifradas) {
    numCifradas = 0;
    bool cifrar = false;
    char c = fOriginal.get();
    while (!fOriginal.eof()) {
        if (c == MARCA_CIFRAR) {
            cifrar = true;
        }
        else if (c == '\n') {
            cifrar = false;
        }

        if (cifrar) {
            c = cifrarCesar(c, clave, numCifradas);
        }
        fCifrado.put(c);
        c = fOriginal.get();
    }
}


/*
 * Pre:  clave >= 0.
 * Post: Si «nombreOriginal» representa el nombre de un fichero de texto
```

```
 *       del que se puede leer y si «nombreCifrado» representa el nombre
 *       de un fichero, Ha copiado el contenido del fichero denominado
 *       «nombreOriginal» en un fichero de texto denominado
 *       «nombreCifrado», aplicando un cifrado César con un desplazamiento
 *       igual al valor del parámetro «clave» solo a aquellos caracteres de
 *       cada línea que aparecén detrás del carácter «MARCA_CIFRAR», ha
 *       asignado al parámetro «numCifradas» el número de letras que se han
 *       copiado a «fOriginal» aplicándoseles el cifrado mencionado y ha
 *       devuelto «true». Si no ha podido leer del fichero denominado
 *       «nombreOriginal» o no ha podido escribir un fichero de texto
 *       denominado «nombreCifrado», ha escrito un mensaje de error en la
 *       pantalla a través de «cerr» y ha devuelto false.
 */
bool cifrarFichero(const char nombreOriginal[], const char nombreCifrado[],
                   const int clave, int& numCifradas) {
    ifstream fOriginal(nombreOriginal);
    if (fOriginal.is_open()) {
        ofstream fCifrado(nombreCifrado);
        if (fCifrado.is_open()) {
            cifrarFichero(fOriginal, fCifrado, clave, numCifradas);
            fCifrado.close();
        }
        else {
            cerr << "No ha podido crearse el fichero \""
                 << nombreCifrado << "\"" << endl;
                 return false;
        }
        fOriginal.close();
    }
    else {
        cerr << "No se ha podido acceder al fichero \"" << nombreOriginal
             << "\"" << endl;
             return false;
    }
    return true;
}

/*
 * Programa que solicita al usuario el nombre de dos ficheros. El primero debe
 * existir y puede contener cualquier tipo de texto. El programa copia el
 * contenido del primer fichero en el segundo, aplicando el cifrado ROT13 a
 * aquellos caracteres de cada línea que se encuentren detrás de un
 * carácter '>'. Los caracteres de las líneas en las que no aparece el
 * carácter '>' se copian sin aplicar ningún cifrado, al igual que los
```

```
 * caracteres que aparecen antes del carácter '>' en las líneas que sí lo
 * contienen.
 * Si no hay ningún problema con la apertura y creación de los ficheros
 * involucrados, el programa informa de que ha podido generar el nuevo fichero
 * correctamente e indica, además el número de letras que se han cifrado en el
 * proceso. En el caso de que el programa no pueda abrir el primer fichero o
 * no pueda crear el segundo, se limita a escribir un mensaje de error.
 */
int main() {
    cout << "Escriba el nombre del fichero a leer: ";
    char nombreFicheroOrigen[MAX_LONG_NOMBRE_FICHERO];
    cin.getline(nombreFicheroOrigen, MAX_LONG_NOMBRE_FICHERO);

    cout << "Escriba el nombre del fichero a escribir: ";
    char nombreFicheroDestino[MAX_LONG_NOMBRE_FICHERO];
    cin.getline(nombreFicheroDestino, MAX_LONG_NOMBRE_FICHERO);

    int numCifradas;
    if (cifrarFichero(nombreFicheroOrigen, nombreFicheroDestino,
                      CLAVE_ROT13, numCifradas)) {
        cout << "El fichero \"" << nombreFicheroDestino
             << "\" se ha generado correctamente." << endl;
        cout << "Se han cifrado un total de " << numCifradas << " letras."
             << endl;
        return 0;
    }
    else {
        return 1;
    }
}
```

Una solución del problema propuesto en el turno 2.º

```
/*****************************************************************************\
 * Curso de Programación 1. Examen práctico de enero de 2020
 * Autores: Miguel Ángel Latre
 * Última revisión: 27-2-2020
 * Resumen: Solución al problema planteado en el turno 2
\*****************************************************************************/
#include <iostream>
#include <fstream>
#include <cctype>
using namespace std;

const int MAX_LONG_NOMBRE_FICHERO = 200;
const int MAX_LONG_LINEA = 200;
const int CLAVE_ROT5 = 5;
const int NUM_DIGITOS = 10;


/*
 * Pre:  ---
 * Post: Si el valor del parámetro «c» es un dígito (base 10), ha devuelto el
 *       dígito correspondiente a aplicar un cifrado ROT5. En caso de que
 *       «c» no sea un dígito, ha devuelto «c» sin hacer ninguna
 *       transformación.
 */
char cifrarROT5(const char c) {
    if (isdigit(c)) {
        int posicion = c - '0';
        int posicionCifrada = (posicion + CLAVE_ROT5) % NUM_DIGITOS;
        return posicionCifrada + '0';
    }
    else {
        return c;
    }
}


/*
 * Pre:  «fOriginal» y «fCifrado» están abiertos.
 * Post: Ha cifrado el contenido pendiente de leer de «fOriginal»
 *       aplicando un cifrado ROT5 y lo ha escrito en «fCifrado». Ha asignado
 *       a «lineasModificadas» el número de líneas en las que se ha cifrado al
```

```
 *       menos un carácter.
 */
void cifrarFichero(istream& fOriginal, ostream& fCifrado,
                   int& numLineasModificadas) {
    numLineasModificadas = 0;
    bool lineaModificada = false;
    char c = fOriginal.get();
    while (!fOriginal.eof()) {
        if (isdigit(c)) {
            lineaModificada = true;
        }
        else if (c == '\n' && lineaModificada) {
            numLineasModificadas++;
            lineaModificada = false;
        }
        fCifrado.put(cifrarROT5(c));
        c = fOriginal.get();
    }
}


/*
 * Pre:  ---
 * Post: Si «nombreOriginal» representa el nombre de un fichero de texto
 *       del que se puede leer y si «nombreCifrado» representa el nombre
 *       de un fichero, ha escrito el contenido del fichero denominado
 *       «nombreOriginal» en un fichero de texto denominado
 *       «nombreCifrado», cifrándolo aplicando ROT5, ha asignado a
 *       «lineasModificadas» el número de líneas en las que se ha cifrado al
 *       menos un carácter y ha devuelto «true».
 *       Si no ha podido leer del fichero denominado «nombreOriginal»
 *       o no ha podido escribir un fichero de texto denominado
 *       «nombreCifrado», ha escrito un mensaje de error en la pantalla a
 *       través de «cerr» y ha devuelto false.
 */
bool cifrarFichero(const char nombreOriginal[], const char nombreCifrado[],
                   int& lineasModificadas) {
    ifstream fOriginal(nombreOriginal);
    if (fOriginal.is_open()) {
        ofstream fCifrado(nombreCifrado);
        if (fCifrado.is_open()) {
            cifrarFichero(fOriginal, fCifrado, lineasModificadas);
            fCifrado.close();
        }
```

```
        else {
            cerr << "No␣ha␣podido␣crearse␣el␣fichero␣\""
                 << nombreCifrado << "\"" << endl;
                 return false;
        }
        fOriginal.close();
    }
    else {
        cerr << "No␣se␣ha␣podido␣acceder␣al␣fichero␣\"" << nombreOriginal
             << "\"" << endl;
             return false;
    }
    return true;
}

/*
 * Programa que solicita al usuario los nombres de dos ficheros de texto. El
 * primero debe existir y puede contener cualquier tipo de texto. El programa
 * copia el contenido del primer fichero en el segundo, aplicando el cifrado
 * ROT5 a los dígitos que se encuentren en el mismo. Los caracteres que no
 * sean dígitos, se copian en el segundo fichero tal cual, sin hacer ningún
 * tipo de cifrado ni de modificación.
 * Si no hay ningún problema con la apertura y creación de los ficheros
 * involucrados, el programa informa de que ha podido generar el nuevo fichero
 * correctamente e indica, además, el número de líneas del fichero en las que
 * ha cifrado algún carácter. En el caso de que el programa no pueda abrir el
 * primer fichero o no pueda crear el segundo, se limita a escribir un mensaje
 * de error.
 */
int main() {
    cout << "Escriba␣el␣nombre␣del␣fichero␣a␣leer:␣";
    char nombreFicheroOrigen[MAX_LONG_NOMBRE_FICHERO];
    cin.getline(nombreFicheroOrigen, MAX_LONG_NOMBRE_FICHERO);

    cout << "Escriba␣el␣nombre␣del␣fichero␣a␣escribir:␣";
    char nombreFicheroDestino[MAX_LONG_NOMBRE_FICHERO];
    cin.getline(nombreFicheroDestino, MAX_LONG_NOMBRE_FICHERO);

    int cifradas = 0;
    if (cifrarFichero(nombreFicheroOrigen, nombreFicheroDestino, cifradas)) {
        cout << "El␣fichero␣\"" << nombreFicheroDestino
             << "\"␣se␣ha␣generado␣correctamente." << endl;
        cout << "Se␣han␣modificado␣un␣total␣de␣" << cifradas << "␣líneas."
             << endl;
```

```
        return 0;
    }
    else {
        return 1;
    }
}
```

Criterios de corrección de los exámenes prácticos

- Compilación (establece un factor de corrección en la nota)
  - Si compila correctamente, el factor es 1.
  - Si compila con advertencias, el factor está entre 0,7 o 0,9, en función de la gravedad de las mismas.
  - Si hay un error de compilación debido a firmar el trabajo en la cabecera sin hacerlo como comentario C++, el factor es 0,9.
  - Algunos errores de compilación y de ejecución tienen un factor de 0,7:
    - Fichero entregado de nombre distinto a «turno1.cpp» o «turno2.cpp»
    - Presencia de cláusulas de inclusión innecesarias de ficheros no solicitados
    - Errores de ejecución por modificar (anteponiendo rutas) el nombre de los ficheros introducidos por el usuario
  - Otro tipo de errores de compilación tienen un factor de corrección de 0.
- Ejecución (50%)
  - Se han realizado 5 pruebas:
    - Descifrando los ficheros «chiste-rot13.txt» (columna T1) y «martin-rot13.txt» (columna T2) en el caso del turno 1 (ver el final del archivo).
    - Descifrando los ficheros «primos-rot5.txt» (columna T1) y «unizar-rot5.txt» (columna T2) en el caso del turno 2 (ver el final del archivo).
    - Columna T3: cifrar de nuevo el fichero resultante de la prueba T2 y comprobar que es idéntico a «martin-rot13.txt» o «unizar-rot5.txt», dependiendo del turno
    - Columna T4: se ha probado con la entrada «inexistente.txt», que no corresponde con ningún fichero que exista, como primer nombre de fichero.
    - Columna T5: se ha probado con un nombre de fichero que sí existe como primer nombre de fichero y con la entrada «a\b.txt» como segundo, no correspondiendo «a» con el nombre de ningún directorio del directorio de ejecución (se ha ejecutado en Windows 10).
- Diseño (50%)
  - Se han valorado los siguientes aspectos:
    - Especificaciones (30%)
    - Diseño (30%)
    - Modularidad (20%)
    - Utilización de constantes simbólicas y no literales, en particular los relativos a la clave (10%)
    - Tabulación y legibilidad (10%)

Programación 1

Examen práctico. 30 de enero de 2020

Contenido original	Contenido descifrado	Letras/líneas cifradas
«chiste-rot13.txt»		
HAY 10 TIPOS DE PERSONAS: > N: YBF DHR FNORA OVANEVB > O: YBF DHR AB	HAY 10 TIPOS DE PERSONAS: > A: LOS QUE SABEN BINARIO > B: LOS QUE NO	28 letras
«martin-rot13.txt»		
CUANDO GEORGE R. R. MARTIN PUBLIQUE LAS NOVELAS QUE FINALIZAN LA SERIE _CANCIÓN DE HIELO Y FUEGO_, EN LA QUE SE BASA LA CONOCIDA SERIE DE TELEVISIÓN _JUEGO DE TRONOS_, > PBZCEBONERZBF FV UN FVQB > PNCNM QR WHAGNE QR AHRIB N YBF CREFBANWRF RA NYTÚA FVGVB. > PBA HA AÚZREB >= DHR 2, ABF CBQEÍNZBF QNE PBA HA PNAGB RA YBF QVRAGRF. UN EXFAN ANÓMINO.	CUANDO GEORGE R. R. MARTIN PUBLIQUE LAS NOVELAS QUE FINALIZAN LA SERIE _CANCIÓN DE HIELO Y FUEGO_, EN LA QUE SE BASA LA CONOCIDA SERIE DE TELEVISIÓN _JUEGO DE TRONOS_, > COMPROBAREMOS SI HA SIDO > CAPAZ DE JUNTAR DE NUEVO A LOS PERSONAJES EN ALGÚN SITIO. > CON UN NÚMERO >= QUE 2, NOS PODRÍAMOS DAR CON UN CANTO EN LOS DIENTES. UN EXFAN ANÓMINO.	115 letras
«primos-rot5.txt»		
Los primeros números primos son los siguientes: 7, 8, 0, 2, 66, 68, 62, 64, 78, 74, 86, 82, 96, 98, 92, 08, 04, 16, 12, 26, 28, 24, 38, 34, 42	Los primeros números primos son los siguientes: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97	1 línea
«unizar-rot5.txt»		
Los datos de contacto de la Universidad de Zaragoza son: Pedro Cerbuna 67 05554 Zaragoza - España Tel: 421 21 65 55 ciu@unizar.es Su CIF es Q-0563556-G	Los datos de contacto de la Universidad de Zaragoza son: Pedro Cerbuna 12 50009 Zaragoza - España Tel: 976 76 10 00 ciu@unizar.es Su CIF es Q-5018001-G	4 líneas