Encriptador y Desencriptador

Usando el lenguaje de programación C++ sin mayor razón más que tener conocimiento práctico en el ámbito de archivos, se codificara un encriptador junto con su desencriptador de archivos de texto plano. Este cifrado estará usando una versión dinámica de Cesar con 2 llaves, y aplicando también Xor con otra llave, para obtener al final una cadena de caracteres de 0 y 1 los cuales representaran la información cifrada. Las llaves se obtendrán a partir de la cantidad de caracteres en el texto, de manera que cambiara con cada texto.

DESARROLLO

El cifrador lo codifique como una Clase, pues en mi planteamiento es un paralelismo con una maquina encriptador, donde metes una hoja con información, y te regresa la información encriptada, de esta manera, esta clase tendrá solo 2 métodos públicos, encriptar y des encriptar.

El seudocódigo que me plantee para poder realizar el algoritmo del cifrado fue el siguiente:

- Obtener valores ASCII de cada carácter en la cadena.
- Guardar los valores en un arreglo.
- La llave 1 será el valor del tamaño modular 7 (0 al 7)
- La llave 2 será el valor del tamaño modular 11 (0 al 11 sin repetir el anterior por ser números primos)
- Para cada valor, exceptuando el valor de las 2 llaves
    - Si la posición del carácter es par usar la llave 1
        - Sumarle la llave 1 al valor, pero si este excede los 255, restarla en vez de sumarla.
    - Si la posición del carácter es impar usar la llave 2
        - Sumarle la llave 2 al valor, pero si este excede los 255, restarla en vez de sumarla.
- La llave 3, o llave Xor, será el total de caracteres % 255, para así evitar que sea mayor a 255, y convertirlo a binario.
- Crear un arreglo de valores en binario.
- Para cada valor.
    - Guardar el valor en binario en el arreglo binario.
    - Aplicar el xor con la llave 3 al valor binario.
    - Añadirle un delimitador.
- Guardar todos los valores binarios en una cadena y regresarla.

De esta manera, en el método de encriptar, recibo una cadena de caracteres, cambio las vocales con tilde, por su equivalente sin ellos porque puede causar mala codificación por usar código ASCII, después, la traduzco uno por uno a su valor en ASCII, guardándolos en un arreglo.

```
string Cifrador::ascii(string aux)
{
    stringstream ss;
    string msg="";
    int i=0;
    while(aux[i])
    {
        string sa;
        int ia=int(aux[i]);
        ss.str("");
        ss.clear();
        ss<<ia;
        sa=ss.str();
        msg+=sa+" ";
        i++;
    }
    return msg;
}
```

```
int datos[tam];
int j=0;
i=0;
while(data[i])
{
    if(data[i]==' ')
    {
        ss.clear();
        ss.str();
        ss<<aux;
        ss>>datos[j];
        j++;
        aux="";
    }
    else
    {
        aux+=data[i];
    }

    i++;
}
```

Consigo las 2 llaves, y las empiezo a aplicar en un for, con las condiciones para cada valor.

```
int key_1=datos[tam%7],key_2=datos[tam%11];
for(i=0;i<tam;i++)
{
    if(i == tam%7 || i == tam%11)
    {
        //...
    }
    else if((i+tam)%2==0)
    {
        if((key_1+datos[i])>255)
            datos[i]-=key_1;
        else
            datos[i]+=key_1;

    }
    else
    {
        if((key_2+datos[i])>255)
            datos[i]-=key_2;
        else
            datos[i]+=key_2;
    }

}
```

Obtenemos las 3ra llave, y con ella aplicamos xor, obteniendo primero cada valor desde entero a binario, rellenando a la izquierda los 0, para evitar perdida de información.

```
string key_xor=toBinaryRev(tam%255);
string bin_datos[tam];
for(i=0;i<tam;i++)
{
    bin_datos[i]=toBinaryRev(datos[i]);
    j=0;
    while(bin_datos[i][j])
    {
        if(bin_datos[i][j]==key_xor[j])
        {
            bin_datos[i][j]='0';
        }
        else
        {
            bin_datos[i][j]='1';
        }
        j++;
    }
    res+=bin_datos[i]+"2";
}
```

```cpp
string Cifrador::toBinaryRev(int n)
{
    std::string r;
    while(n!=0) {r=(n%2==0 ?"0":"1")+r; n/=2;}
    int i=0,zeros=8;
    while(r[i])
    {
        i++;
    }
    zeros=zeros-i;
    while(zeros>0)
    {
        r="0"+r;
        zeros--;
    }
    return r;
}
```

Teniendo el arreglo, añadimos cada valor junto con un delimitador a un string, y así regresamos la información encriptada. De esta manera

“110000012101110102101000102101000112101110002111011102101101112000000112101001102101110002111000112101101112101110112101001012101101112000011102111000112101011012101100012101000012111000112101011012101111112101110102111000112110001012111000112101100112101001102101000012111000112101001012101111012111011102101111102101101012111000112101001012101111012101000002101000102101110002101000002101101012000110012110111002101001102101000102111000112101011012101111012101000102111000112101101012101001102101110102101111112101111112101101002111011102101101112101001102101001102101110002111000112101000112101000102101110012101001102111011102101101112101111112111000112101011002101001102111110102111000112101111002101001102101011012101000002101001102101110102101110002101001002111011102101111002101100012101101112111011102101101112101111112111000112101110012101001102000001002110111112101000012101101112101100112111000112101110012101001102111011102101100002101000012101001102”

Aunque alguien obtenga los caracteres ASCII a través del binario, serán caracteres aleatorios, que no tendrán un patrón fácilmente detectable. La razón por la que mantuve en binario los datos fue que no sea para nada legible a la vista humana.

Las llaves al ser dinámicas y depender del archivo en cuestión, hará que un descifrado humano no sea nada fácil, ya que además depende de cómo el lenguaje maneja los decimales a enteros, es por eso que use división para obtener las llaves.

Como dato extra, este cifrador se puede mandar a llamar recursivamente las rondas que se desee, siendo que aumenta * 8 los caracteres cada vez que se cifra, complicando aún más un descifrado humano. Aunque realmente no aplico más de 1 ronda, para mantenerlo en su estado más puro y que la persona que lo cifra pueda decidir qué tan ilegible sea su contenido, además de que al tener del cifrador, realmente no sería un problema rondas extra.

Tenemos el texto cifrado, a continuación se tendrá que programar el desencriptador, recibiendo la cadena de bits.

Este hará ciertamente los pasos del encriptador a la inversa, siguiendo el siguiente seudocódigo:

- Obtener los valores binarios de la cadena, usando el delimitador, y guardarlos en un arreglo de valores binarios.
- La llave 3, o llave xor, será el total de caracteres % 255, para así evitar que sea mayor a 255, y convertirlo a binario.
- Crear un arreglo de valores en binario.
- Para cada valor binario.
    - Aplicar el xor con la llave 3 al valor binario.
- Crear un arreglo de valores.
- Para cada valor.
    - Guardar el valor binario en los valores enteros
- La llave 1 será el valor del tamaño modular 7 (0 al 7)
- La llave 2 será el valor del tamaño modular 11 (0 al 11 sin repetir el anterior por ser números primos)
- Para cada valor, exceptuando el valor de las 2 llaves
    - Si la posición del carácter es par usar la llave 1
        - Sumarle la llave 1 multiplicada por 2 al valor, si este excede los 255, restar la llave, pero si no lo hace, sumarla.
    - Si la posición del carácter es impar usar la llave 2

        - Sumarle la llave 2 multiplicada por 2 al valor, si este excede los 255, restar la llave, pero si no lo hace, sumarla.
- Obtener el valor ASCII para cada valor entero.
- Regresar la cadena de caracteres que estará des encriptada y legible.

Como se puede notar, el seudocódigo es prácticamente el mismo, a excepción de la partes de las llaves 1 y 2.

Se obtiene los valores binarios desde la cadena de caracteres.

```
string bin_datos[tam];
string binaux="";
while(data[i])
{
    if(data[i]=='2')
    {
        bin_datos[j]=binaux;
        binaux="";
        j++;
    }
    else
        binaux+=data[i];
    i++;
}
```

Se vuelve a aplicar el xor con la llave 3, para recuperar los datos originales, y se obtiene el valor entero de cada valor binario, guardado en otro arreglo.

```
int datos[tam];
string key_xor=toBinaryRev(tam%255);
for(i=0;i<tam;i++)
{
    j=0;
    while(bin_datos[i][j])
    {
        if(bin_datos[i][j]==key_xor[j])
        {
            bin_datos[i][j]='0';
        }
        else
        {
            bin_datos[i][j]='1';
        }
        j++;
    }
}

for(i=0;i<tam;i++)
{
    datos[i]=binarioToDecimal(bin_datos[i]);
}
```

Este internamente tiene un método privado para obtener un decimal a través de un binario.

```
int Cifrador::binarioToDecimal(string aux)
{
    stringstream ss;
    int n;
    ss.clear();
    ss.str();
    ss<<aux;
    ss>>n;
    int decimalNumber = 0, i = 0, remainder;
    while (n!=0)
    {
        remainder = n%10;
        n /= 10;
        decimalNumber += remainder*pow(2,i);
        ++i;
    }
    return decimalNumber;
}
```

Aplicamos las llave 1 y 2 como en el seudocódigo de des encriptación. Esta será la única parte de código que será ligeramente diferente en la encriptación y des encriptación.

```
int key_1=datos[tam%7],key_2=datos[tam%11];
for(i=0;i<tam;i++)
{
    if(i == tam%7 || i == tam%11)
    {
        //...
    }
    else if((i+tam)%2==0)
    {
        if(datos[i]+(key_1*2)>255)
            datos[i]-=key_1;
        else
            datos[i]+=key_1;

    }
    else
    {
        if(datos[i]+(key_2*2)>255)
            datos[i]-=key_2;
        else
            datos[i]+=key_2;
    }


}
```

Así teniendo nuestro arreglo de números ASCII como valores reales, los convertimos a carácter, y los regresamos como una cadena de caracteres.

```
string res="";
for(i=0;i<tam;i++)
{
    char c;
    c=datos[i];
    res+=c;
}

return res;
```

“Black then white are all I see in my infancy

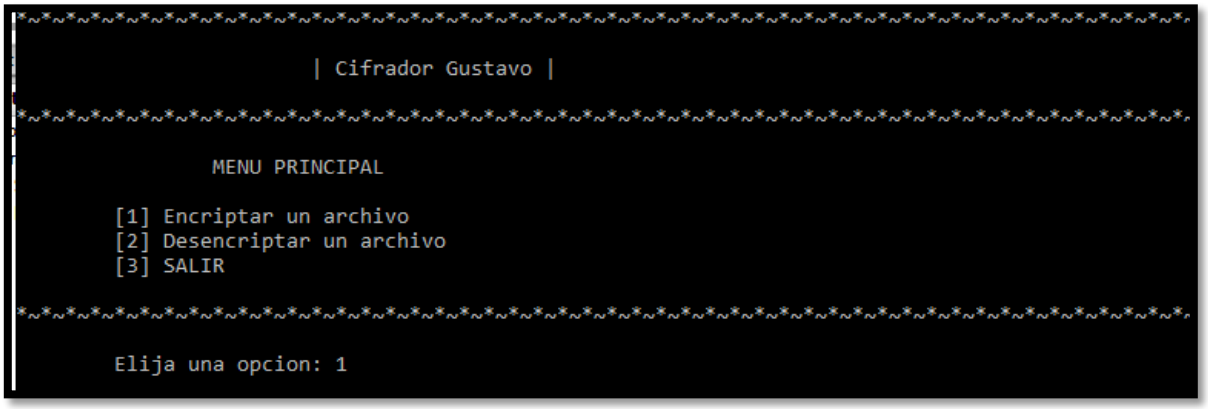
Red and yellow then came to be, reaching out to me

Lets me see”

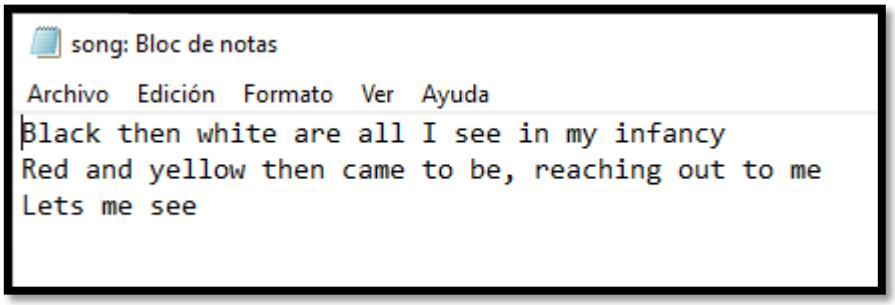
Así, simplemente implementando una interfaz, que nos deje cargar un archivo obteniendo todo su contenido y guardándolo en un string, podemos mandar a llamar los métodos de la clase Cifrador, para obtener el encriptamiento y desencriptamiento.

CAPTURAS

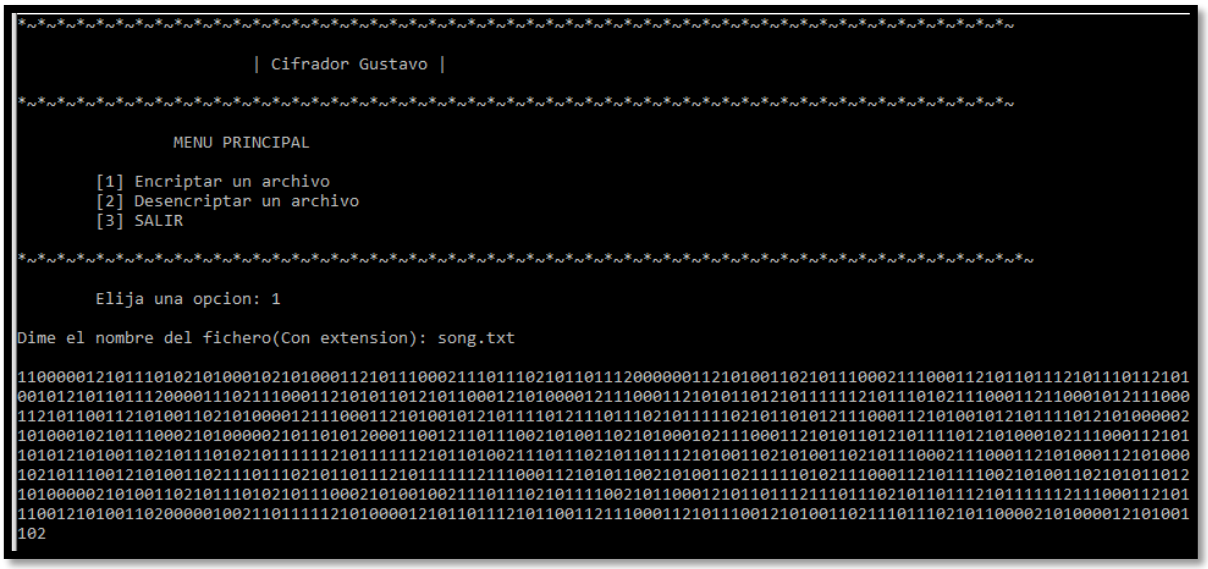
Menú principal:



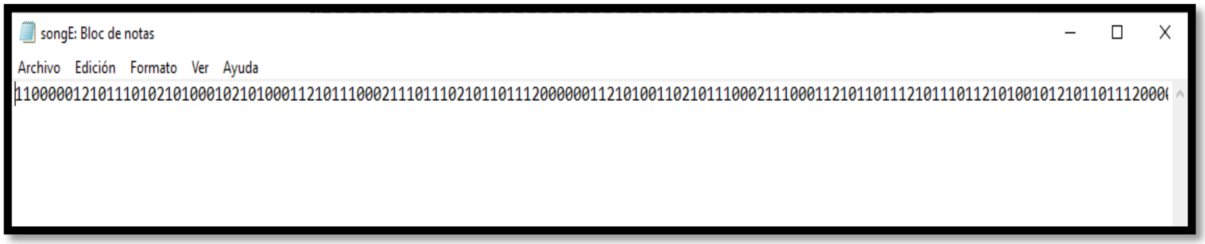
Archivo original:



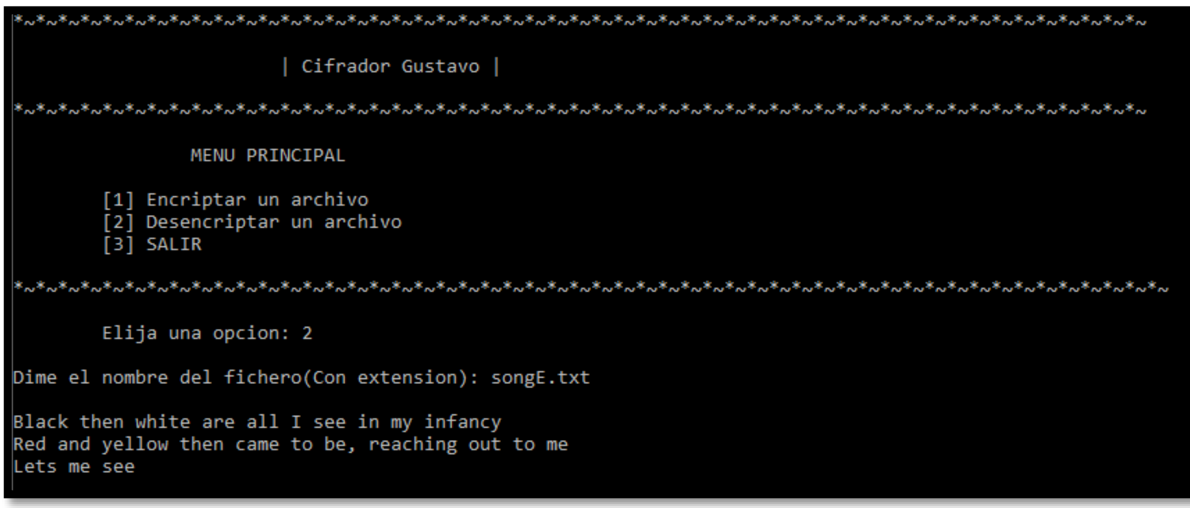
Encriptado de un archivo:



Archivo encriptado guardado:



Des encriptado de un archivo:



Archivo des encriptado guardado: