ÍNDICE

1. INTRODUCCIÓN

En esta actividad se implementa una variante del Método César de encriptación.

El programa da la opción tanto de encriptar como de desencriptar un mensaje eligiendo una clave numérica que será usada en el algoritmo criptográfico.

En este programa se ha limitado la longitud del mensaje a 25 caracteres y el rango del valor de la clave, que podrá ser entre 0 y 200.

El algoritmo de encriptación consiste en lo siguiente:

- Primero se realizan los filtros en la entrada de datos iniciales (la clave y el tamaño del mensaje).
- Después se lee el mensaje carácter a carácter transformándolo en su valor encriptado. En este caso consiste en la suma del valor entero de la tabla ASCII de dicho carácter y del valor de la clave.
- Seguidamente se muestra en pantalla el valor encriptado y se va almacenando el texto unido en una cadena de caracteres.

Este método de encriptado es bastante débil. Permite el uso de las frecuencias en que suelen aparecen las letras del abecedario en cada idioma para descubrir la clave usada y así recuperar el mensaje original de una forma fácil.

En una segunda fase de la actividad se ha implementado un sistema de comprobación de lo anterior. Para ello, cuando se van recogiendo los caracteres del texto se va calculando la frecuencia de aparición de las 7 letras más habituales en castellano. Al terminar se muestra la frecuencia de aparición de cada una de esas 7 letras para su cotejo.

Para evitar salvar un poco la debilidad del método anterior he realizado una tercera fase que viene explicada en el apartado 4.

Seguidamente presento el código fuente y las pruebas realizadas.

2. CÓDIGO FUENTE DEL PROGRAMA EN C++

```
/*
 * AEC 1 - Práctica 1 de programación con C++
 * Realizado por Juan del Rio Huertas
 */

/*
 * Programa para encriptar o desencriptar mensajes de texto.
 * Está basado en el Método César pero sin realizar una asignación circular
 * de los códigos y sólo asignando un código numérico
 *
 */

#include <iostream>

using namespace std;

void encriptar(void);
```

```
void desencriptar(void);
bool modo_seguro;
/*
 *
 */
int main(void)
{
    modo_seguro = false;
    cout << "Encriptador por el Método César" << endl;
    int opcion_menu;
    bool salir= false;

    // Mostrar menu de opciones ----------------------------------
    // Añado una característica adicional: Modo "Seguro"
    // cuando se activa el algoritmo de encriptación se fortalece un poco
    // ya que dependerá de:
    //   - clave
    //   - posición del cáracter en el texto
    //   - longitud del texto

    do
    {

        cout << "0) ¿Desea activar modo \"seguro\"? (S/N)" << endl;
        cout << "1) Introducir texto para encriptar." << endl;
        cout << "2) Introducir texto en clave para desencriptar." << endl;
        cout << "3) Salir del programa." << endl << endl;
        cin >> opcion_menu;

        switch (opcion_menu)
        {
            case 0:

                if (modo_seguro){
                    modo_seguro = false;
                    cout << "Modo \"seguro\" desactivado. " << endl;
                }else{
                    modo_seguro = true;
                    cout << "Modo \"seguro\" activado. " << endl;
                }
                break;
            case 1: encriptar(); break;
            case 2: desencriptar(); break;
            case 3:
                cout << "Se cerrará la aplicación.\nHasta la próxima. Gracias por intentarlo." <<
endl ;
                salir = true; break;
            default: cout << "Opción no disponible. Pruebe otra vez" << endl; break;
        }
    }while (!salir);


    return 0;
}

// Encripta mensaje con un máximo de 25 letras usando una clave numérica entre 0 y 200

void encriptar(void)
{
   char letra_a_leer = ' ';
   int codigo_de_letra_encriptada = 0;
   string texto_a_encriptar = "";
   int longitud_texto;
   int clave;
   bool longitud_incorrecta_texto = true;
   bool longitud_incorrecta_clave = true;

   int contador_e= 0;
   int contador_a= 0;
   int contador_o= 0;
   int contador_s= 0;
   int contador_r= 0;
   int contador_n= 0;
   int contador_i= 0;
```

```
   // se aplican los filtros
   do
   {
        cout << "Introduce la longitud del texto a encriptar (1-25 letras)" << endl;
        cin >> longitud_texto;

        longitud_incorrecta_texto = (longitud_texto < 1 || longitud_texto > 25 );

        if (cin.fail()){  // filtro para el caso de que se introduzca un valor no numérico
            cout<< "Error. Introduce un valor numérico\n";
            cin.clear();
            cin.ignore(2, '\n');
        }else if (longitud_incorrecta_texto == true){ // filtro para el caso de que la longitud no
se adecuada
            cout << "Longitud incorrecta." << endl;
        }

    }while (longitud_incorrecta_texto);

    do
    {
       cout << "Introduce la clave para encriptar (numero entre 0 a 200): " ;
       cin >> clave;
       longitud_incorrecta_clave = (clave < 0 || clave > 200 || cin.fail());
       if (cin.fail()){
            cout<< "Error. Introduce un valor numerico\n";
            cin.clear();
            cin.ignore(2, '\n');
        }else if (longitud_incorrecta_clave == true){
            cout << "Longitud incorrecta clave." << endl;
        }
cout <<longitud_incorrecta_clave ;
   }while (longitud_incorrecta_clave );



   for (int i=0;i<longitud_texto;i++)
   {
       cout << "Introduce el caracter numero " << i+1 << " del texto: " ;
       cin >> letra_a_leer;

       // Implementación de la fase 2: calcula la frecuencia de aparición
       // de las 7 letras más habituales en castellano.
       switch (letra_a_leer)
       {
           case 'e': contador_e++;break;
           case 'a': contador_a++;break;
           case 'o': contador_o++;break;
           case 's': contador_s++;break;
           case 'r': contador_r++;break;
           case 'n': contador_n++;break;
           case 'i': contador_i++;break;
           case 'E': contador_e++;break;
           case 'A': contador_a++;break;
           case 'O': contador_o++;break;
           case 'S': contador_s++;break;
           case 'R': contador_r++;break;
           case 'N': contador_n++;break;
           case 'I': contador_i++;break;

       }

       // se aplica el modo seguro en caso de estar activado

       if (modo_seguro){
           codigo_de_letra_encriptada= (int)letra_a_leer + (clave+1) + (i+1) + longitud_texto  ;
       }else{
        codigo_de_letra_encriptada= (int)letra_a_leer + clave;
       }

       cout << "- Codigo para este carácter: " << codigo_de_letra_encriptada << endl;

       texto_a_encriptar = texto_a_encriptar + letra_a_leer;
```

```
   }
   cout << "Fin del texto" << endl;
   cout << "Texto completo:" << texto_a_encriptar << endl << endl;
   cout << "ESTADISTICA:" << endl ;
   cout << "La letra E aparece " << contador_e << " veces, un " << (float)contador_e/
longitud_texto * 100 << " %" << endl ;
   cout << "La letra A aparece " << contador_a << " veces, un " << (float)contador_a/
longitud_texto * 100 << " %" << endl ;
   cout << "La letra O aparece " << contador_o << " veces, un " << (float)contador_o/
longitud_texto * 100 << " %" << endl ;
   cout << "La letra S aparece " << contador_s << " veces, un " << (float)contador_s/
longitud_texto * 100 << " %" << endl ;
   cout << "La letra R aparece " << contador_r << " veces, un " << (float)contador_r/
longitud_texto * 100 << " %" << endl ;
   cout << "La letra N aparece " << contador_n << " veces, un " << (float)contador_n/
longitud_texto * 100 << " %" << endl ;
   cout << "La letra I aparece " << contador_i << " veces, un " << (float)contador_i/
longitud_texto * 100 << " %" << endl << endl ;

}

// Desencripta mensaje con un máximo de 25 letras usando una clave numérica entre 0 y 200

void desencriptar(void)
{
   int codigo_a_leer = 0;
   char caracter_desencriptado = ' ';
   string texto_desencriptado = "";
   int longitud_texto;
   int clave;
   bool longitud_incorrecta_texto = true;
   bool longitud_incorrecta_clave = true;

   // se aplican los filtros
   do
   {
        cout << "Introduce la longitud del texto a encriptar (1-25 letras)" << endl;
        cin >> longitud_texto;
        longitud_incorrecta_texto = (longitud_texto < 1 || longitud_texto > 25);
        if (cin.fail()){
            cout<< "Error. Introduce un valor numérico\n";
            cin.clear();
            cin.ignore(2, '\n');
        }else if (longitud_incorrecta_texto == true){
            cout << "Longitud incorrecta." << endl;
        }
    }while (longitud_incorrecta_texto);

    do
    {
       cout << "¿Cual era la clave para desencriptar? (numero entre 0 a 200): " ;
       cin >> clave;
       longitud_incorrecta_clave = (clave < 0 || clave > 200 || cin.fail());
       if (cin.fail()){
            cout<< "Error. Introduce un valor numérico\n";
            cin.clear();
            cin.ignore(2, '\n');
        }else if (longitud_incorrecta_clave == true){
            cout << "Longitud incorrecta clave." << endl;
        }

   }while (longitud_incorrecta_clave );

   for (int i=0;i<longitud_texto;i++)
   {
       cout << "Introduce el codigo del caracter numero " << i+1 << " del texto: " ;
       cin >> codigo_a_leer;

       // se aplica el modo seguro en caso de estar activado
       if (modo_seguro){
           caracter_desencriptado= (int)codigo_a_leer - (clave+1) - (i+1) - longitud_texto ;
       }else{
        caracter_desencriptado= (int)codigo_a_leer - clave;
       }
```

```
        cout << "- Caracter para este código: " << caracter_desencriptado << endl;
        texto_desencriptado = texto_desencriptado + caracter_desencriptado;

    }
    cout << "Fin del texto" << endl;
    cout << "Texto completo:" << texto_desencriptado << endl << endl;

}
```

3. PRUEBAS REALIZADAS

He realizado diferentes pruebas para comprobar el programa:

Por un lado, he introducido diferentes valores en cada una de las entradas de datos, dentro y fuera de los límites para confirmar el buen funcionamiento. Además, he probado a introducir valores no numéricos cuando la entrada requería número. Esto me producía un error y el programa entraba en un bucle infinito. Solucionarlo me ha llevado bastante tiempo y quebradero de cabeza. Al final recurriendo a la referencia de **cin** de ***iostream*** en http://www.cplusplus.com/ he averiguado lo siguiente:

- La función *fail()* devuelve verdadero si la entrada recibida da error.
- La función clear() resetea el estado de error.
- La función *ignore()* descarta valores que se hayan en el canal *cin* y evita que genere el bucle infinito.

Usando las tres funciones he logrado un buen funcionamiento, aislando el error y después empezando de nuevo.

Por otro lado, he realizado pruebas con diferentes textos para comprobar el correcto funcionamiento de encriptado y desencriptado. Por ejemplo:

Texto de entrada: *En_un_lugar_de_la_Mancha* , longitud =24, clave = 100

169,210,195,217,210,195,208,217,203,197,214,195,200,201,195,208,197,195,177,197,210,199,204,197

Estadísticas: E = 8.33% , A= 16.67%, R= 4.16 %, N= 12.5% y el resto (O,S,I) = 0

4. MODIFICACIÓN DEL ALGORITMO CRIPTOGRÁFICO

Meditando sobre la actividad se me ocurrió una mejora que no supone mucha alteración del código pero que aporta mayor seguridad. La he definido como una tercera fase de la actividad.

En esta fase añado la posibilidad de elegir otro método más seguro. Éste sigue siendo bastante débil comparado con otros métodos de criptográficos (como los de clave pública y privada usados en los certificados digitales) pero elimina la posibilidad de usar simplemente la lista de frecuencias comentada en la introducción.

En este método, para el cálculo del código se usan tanto los valores de la clave como el valor de la longitud del texto y la posición que ocupa cada letra en dicho mensaje.

Pueden elegirse muchas fórmulas para el cálculo usando diferentes operadores aritméticos (+,-,*,/ o %), en este caso he elegido una sencilla:

función= valor entero del carácter (ASCII) + (clave+1) + posición del carácter en el texto + longitud del texto

De tal forma que la misma letra generará códigos distintos según esté en una posición distinta del texto y según el tamaño del mensaje. Por ejemplo:

Si se elige la clave 100 y mensaje "CASA" la letra A genera los siguientes códigos:

Método original: 1ª A = 165, 2ª A = **165**
Método modificado: 1ªA = 65+ (101+2+4)= **172** , 2ª A= 65+(101*4*4)=**174**

En el ejemplo del apartado anterior, eligiendo el modo "seguro" los códigos serían:

195,237,223,246,240,226,240,250,237,232,250,232,238,240,235,249,239,238,221,242,256,246,252,246

Puede verse que hay códigos repetidos, pero no representan a la misma letra, por lo que es más difícil descifrarlo.

Si implementásemos el algoritmo del Método Cesar original, dependiendo de la fórmula elegida, la función de la fórmula podría no ser inyectiva. Es decir, distintos caracteres de entrada en la misma posición podrían generar el mismo código o letra, la función de desencriptado no daría un valor único invalidando el método. En nuestro caso, este problema no se produce por no ser circular la asignación de los códigos.