



DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE CONCEPCIÓN
CONCEPCIÓN, CHILE.

Informe Tarea N°1

Profesor: Mario Medina C.

Ayudante: Diego Ramírez V.

*Alumnos: Nicolás Borcoski S.
Aldo Mellado O.*

Esteban Paz F.

24 de mayo de 2018

Índice

| | |
|---------------|---|
| 1. Problema 1 | 1 |
| 2. Problema 2 | 4 |
| 3. Problema 3 | 7 |

1. Problema 1

Para el primer problema se pide crear una escitala, la cual era una técnica de codificación usada en la antigüedad, la cual consistía en un pergamino que era enrollado al rededor de un cilindro de un diámetro conocido por ambas partes.

Para efectos de la tarea, se decía que esta debía depender de la cantidad de vueltas, y además del texto que se le entregara, el cual era introducido por consola.

```
1 string escitala(const vector<string>& texto, const unsigned int vueltas)
2 {
3     vector<string> aux=texto;
4     string text, aux;
```

Para estas líneas, se define una función llamada *escitala*, la cual recibe un vector del tipo *string* el cual es referenciado y no puede ser modificado tal como lo indica *const*, además, esta función recibe una variable que en este caso será el número de vueltas que "daría al cilindro".

También se definió un vector tipo *string* auxiliar sobre el cual copiamos el contenido en *texto*, finalmente se definen dos *strings*; *text*, que es el *string* sobre el cual copiaremos la frase a codificar y a la cual le quitaremos los espacios, y *auxt*, que es el *string* sobre el cual realizaremos la codificación y que es el que devolverá la función.

```
1     int j=vueltas,lent;
2
3     for(auto x:aux){text=text+x;}
4
5     string::iterator it1=text.begin(),it2 =text.begin();
6     advance(it2,j);
7     lent = text.size();
```

Se definen las constantes que son el número de vueltas, y *lent*, el cual se define más tarde como el largo del texto "*EnunlugardelaManchadecuyonombrenoquieroacordarme*" previamente trabajado y almacenado en *text*. Se uso *advance*, para poder mover el iterador desplazandolo en tantas unidades como se defina el largo o el número de vueltas en este caso.

```
1
2     for(it1;it1!=it2;it1++)
3     {
4         for(int k=0;k<lent;k=k+vueltas)
5             {auxt=auxt + *(it1+k);}
6     }
7
8     return aux;
```

En estas líneas se usan dos ciclos *for*, el primer ciclo definido es uno que hace que el iterador se mueva unidad cada vez que se han codificado lados-terminos del *string*, en el caso de que sea *lados = 8*, se moverá una unidad el iterador definido como *it1*, una vez se hayan codificado 7 letras.

Luego, el segundo *for* es aquel que hace que se vaya agregando al *string auxt* el carácter que está desplazado, respecto del iterador, *k* unidades. Al comienzo, aparece *k*, definido como cero, luego se va sumando al número de vueltas, este número nos indica qué posición de carácter se copiará al *string auxt*.

```

1 int main() {
2     unsigned int lados;
3     int lent;
4
5     cout << "Ingrese el numero de lados: ";
6     cin >> lados;
7     cout << "Ingrese la frase a codificar (termine con ^D): ";
8
9     vector<string> texto;
10    string temp;
11
12    while (cin >> temp)
13        {texto.push_back(temp);}
14
15    cout<< "La frase codificada es: " <<escitala(texto,lados)<<endl;
16
17    return 0;
18 }

```

Finalmente en este fragmento de texto se tiene la función `main`, que es en la que se solicita ingresen una frase, el número de lados y se llama a la función *escitala*, imprimiéndose la palabra codificada.

Las consideraciones que se tuvieron que hacer, fueron aquellas asociadas a que para strings no se puede hacer `push_back()`, de modo que se tenían que hacer concatenaciones. Además, como sucede en la línea en que se define la función, se tiene que dado que se definió como `const` el `vector<string>& texto`, se tuvo que hacer una copia de lo contenido en el, por este motivo se creo `vector<string> aux = texto`.

Resultados Problema 1

```

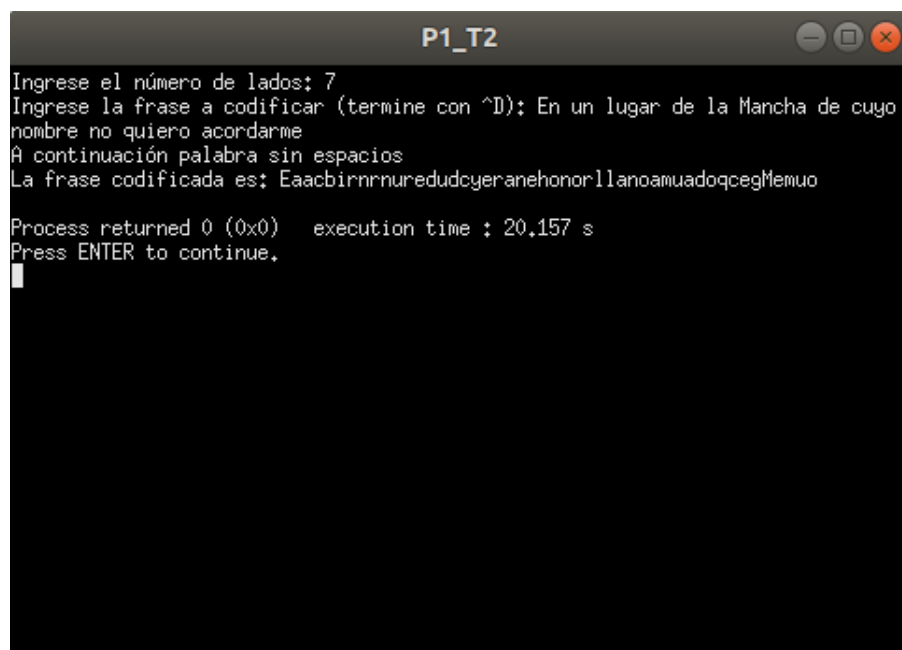
P1_T2
Ingrese el número de lados: 8
Ingrese la frase a codificar (termine con ^D): En un lugar de la Mancha de cuyo
nombre no quiero acordarme
A continuación palabra sin espacios
La frase codificada es: ErcoocndhnqoueaournldmidlaebeauMcrrrgaueomanynae

Process returned 0 (0x0)   execution time : 16.036 s
Press ENTER to continue.

```

Fig. 1: Codificación escitala para *lados* = 8

De la figura 1 se tiene que la codificación entregada se condice con lo esperado y presentado en la guía de laboratorio, pues, lo entregado por la función es un string `ErcoocndhnqoueaournldmidlaebeauMcrrrgaueomanynae`



```
P1_T2
Ingrese el número de lados: 7
Ingrese la frase a codificar (termine con ^D): En un lugar de la Mancha de cuyo
nombre no quiero acordarme
A continuación palabra sin espacios
La frase codificada es: EaacbirnrnuredudcyeranehonorllanoamuadoqcegMemuo

Process returned 0 (0x0)   execution time : 20.157 s
Press ENTER to continue.
```

Fig. 2: Codificación escitala para $lados = 7$

Luego, en la figura 2 se tiene que para el caso de **lados = 7**, el texto recibido se condice con lo esperado pues se obtiene **EaacbirnrnuredudcyeranehonorllanoamuadoqcegMemuo**.

2. Problema 2

Primero, se invita al usuario a elegir si desea codificar o decodificar el archivo entrada.txt. En caso que su elección no sea válida se repite la pregunta.

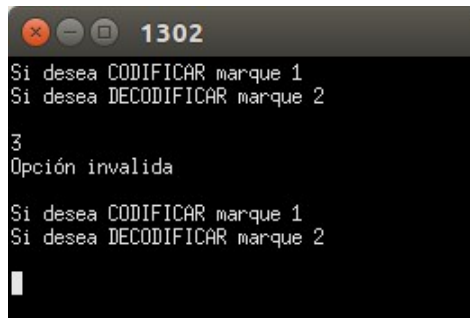


Fig. 3: Mensaje de error

De acuerdo a su elección, luego se consulta por el número de filas y desplazamientos a utilizar.

Los desplazamientos, en caso de ingresarse, se traducen en guiones (-) que se incluyen al inicio del vector de string, lo que vendría a ser algo así:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | A | t | a | q | u | e | a | l | a | m | a | n | e | c | e | r |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Fig. 4: Codificación del mensaje "Ataque al amanecer", filas = 3, desplazamiento = 1

Se pasa el texto de ?entrada.txt? carácter por carácter a un vector de string, en donde cada letra es un string.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | t | a | q | u | e | a | l | m | a | n | e | c | e | r |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Auae -- 4 letras
 tqelmnrcr -- 8 letras
 aaae -- 4 letras

Fig. 5: Codificación del mensaje "Ataque al amanecer", filas = 3, desplazamiento = 0

| | | | | | | |
|---|---|---|---|---|---|---|
| A | | A | | | | e |
| t | | e | l | | n | c |
| a | u | | A | a | | e |
| q | | | m | | | r |

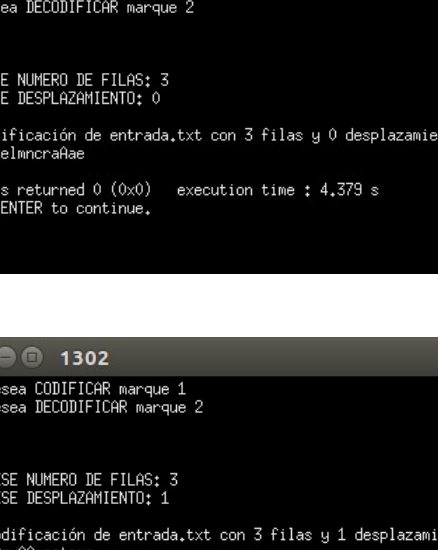
AAe -- 3 letras
 telnc -- 5 letras
 auAae -- 5 letras
 qmr -- 3 letras

Fig. 6: Codificación del mensaje "Ataque al amanecer", filas = 4, desplazamiento = 0

Se puede ver que el primer grupo de letras (la primera fila), su primera letra aparece por primera vez en la posición $2n-2$, siendo n el número de filas y vuelve a aparecer a una distancia de $+(2n-2)$. Así se puede generar una salida con la primera fila. Mediante la función de la STL "fill" se va llenando vector entrada con "/" cuando ya se realizó el "push_back" de esa letra al vector de salida. Se tiene un contador que va guardando la cantidad de letras por cada fila, lo cual se utiliza para el proceso de decodificación.

Se repite lo mismo para las filas entre la primera y la última, con un ciclo iterativo, con la salvedad que en el primero de estos casos caso se realiza un pushback inicial individual y luego se van guardando de a 2 las letras que están seguidas (pensando en una separación por filas).

Tras almacenar en el orden deseado las letras en el vector de salida se remueven mediante la función "remove" los indicadores de desplazamiento y de espacios vacíos. La salida se copia a salida.txt mediante ofstream y se almacena en un vector para facilitar su despliegue en pantalla, el resultado para el ejemplo "Ataque Al Amanecer" presentado en la guía es el siguiente:



```
1302
Si desea CODIFICAR marque 1
Si desea DECODIFICAR marque 2
1
INGRESE NUMERO DE FILAS: 3
INGRESE DESPLAZAMIENTO: 0
La codificación de entrada.txt con 3 filas y 0 desplazamientos es:
AuAetqelmnraAae
Process returned 0 (0x0)    execution time : 4.379 s
Press ENTER to continue.

1302
Si desea CODIFICAR marque 1
Si desea DECODIFICAR marque 2
1
INGRESE NUMERO DE FILAS: 3
INGRESE DESPLAZAMIENTO: 1
La codificación de entrada.txt con 3 filas y 1 desplazamientos es:
qlnrAauAAaeetemc
Process returned 0 (0x0)    execution time : 2.987 s
Press ENTER to continue.
```

Fig. 7: Codificación del mensaje "Ataque al amanecer" mediante código escrito

Ahora para el proceso de decodificación se utiliza el texto entregado en la guía como entrada.txt y se generan vectores que almacenen los grupos de filas, por ejemplo:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | t | a | q | u | e | a | l | m | a | n | e | c | r |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Auae -- 4 letras
 tqelmncr -- 8 letras
 aae -- 4 letras

Fig. 8: Codificación del mensaje "Ataque al amanecer", $\text{filas} = 3$, $\text{desplazamiento} = 0$

Se almacena en un vector de vectores cada vector correspondiente a cada subdivisión del texto de entrada. Se sabe cuántos caracteres guardar en cada vector gracias al contador que corre desde el código del codificador.

Luego, mediante iteradores se recorre cada vector componente del vector que contiene los vectores subdivididos.

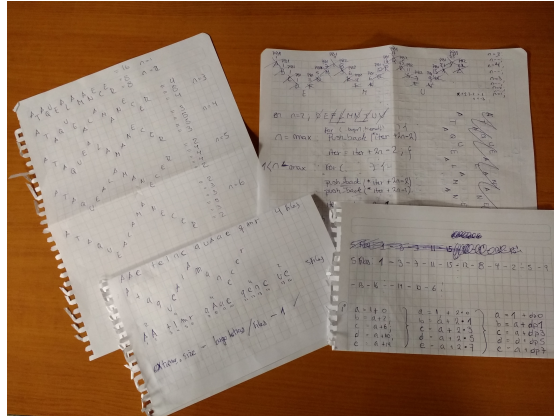


Fig. 9: Apuntes en papel

La idea es recorrer los vectores más pequeños como se haría en papel escribiendo la codificación en una línea continua.

A continuación se presentaron complicaciones para encontrar una forma generalizada de abordar el recorrido de estos vectores, de forma que la cantidad de veces que se repite el proceso tuvo que ser modificada de forma manual mediante "ensayo y error" hasta que no se produjera error de segmentación.

En ese proceso se llegó a la decodificación con 6 filas y 2 desplazamientos que dio el siguiente resultado:

```

1302
Si desea CODIFICAR marque 1
Si desea DECODIFICAR marque 2
2
INGRESE NUMERO DE FILAS: 6
INGRESE DESPLAZAMIENTO: 2

La decodificación de entrada.txt con 6 filas y 2 desplazamientos es:
-CENUNLUGARDELA MANCHADECUYONOMBRENQUIEROACORDARMENOHAMUCHOTIEMPOQUEVIVIAUNHIDAL
CODELOSDLANZAENASTILLERODARGANTIGUAROCINFLACOYCALCOCORREDORUNAOLLADEALCOMASVA
CAQUECARNEROSALPICONLASHASNOCHESDUELOSQUEBRANTOSLOSSABADOSLENTEJASLOSVIERNESALG
UNPALOMINODEANADIDURALOSDOMINGOSCONSUMIANLASTRESPARTESDESUHA
Process returned 0 (0x0)   execution time : 2.471 s
Press ENTER to continue.

```

Fig. 10: Apuntes en papel

En donde se pueden reconocer las primeras líneas de "El Ingenioso Hidalgo Don Quijote de la Mancha" de Miguel de Cervantes.

3. Problema 3

Para desarrollar este problema tuvimos varias ideas. Pero elegimos la siguiente solución porque era la más fácil de implementar a nuestro criterio, aunque en algunos fragmentos del código es más mecánica que ingeniosa.

Codificación: Primero se recibe el archivo del Don Quijote y se insertan a mano guiones bajos, los cuales hacen que no se pierdan los espacios y no se descuadren las coordenadas. Luego este texto es transcrito en una lista con caracteres llamada "pad_ tarea2". Notamos que cada 75 espacios en esta lista se tiene una nueva fila, entonces se escribe una función llamada "coord" en la línea 10. Esta función escribe pares de coordenadas de acuerdo a una relación encontrada con respecto a la posición.

Se genera un objeto Mapa el cual contiene 45 listas, una para cada carácter. (Se sabe que son 45 porque se desarrolla una lista llamada "all_ characters" y se mide el largo y se ve cuáles son los caracteres en el texto).

En cada una de las listas del mapa se escribe la posición de ocurrencia de cada carácter, como se ve entre la línea 78 a la 394 y la llave de cada lista es en efecto el mismo carácter.

Para los casos de K y W se designa por defecto los valores de posición "-2" y "-1" respectivamente y asimismo se ponen condiciones en la función "coord" para que a estos valores asigne los pares (0,0) y (0,1) como fue pedido.

Después de este proceso se genera una función llamada "azar" en la línea 46, la cual toma un elemento al azar de una lista y lo retorna.

Luego se tiene un texto de entrada el cual es el que se quiere codificar, en la línea 408. Este texto "entrada.txt" al igual que se hace con el pad, es transformado en una lista de caracteres llamada "entrada" en la línea 414.

Esta lista "entrada" tiene por valores caracteres, los cuales son asociados a las llaves o keys del mapa, entonces se hace un ciclo que recorra "entrada" y asocie a ella algún valor azaroso de las listas del mapa, entre las líneas 410 a 433.

Luego teniendo las posiciones azarosas del texto entrada ligadas a las posiciones del don Quijote para cada carácter se toman las coordenadas con la función antes dicha "coord.", en la línea 435.

Finalmente se tiene una lista con los pares de coordenadas correspondientes y se imprimen en un texto llamado "salida.txt".

Decodificación:

Se recibe como entrada el mismo texto "salida.txt", y símil a la codificación esta es puesta en una lista de caracteres llamada "salida2", en la línea 451. Nos damos cuenta que tenemos valores en la lista que no sirven, los cuales son los paréntesis y la coma, que servían para denotar las coordenadas de forma bonita, así que estos son removidos de la lista, en las líneas 455 a 457.

Luego esta lista es transformada de contener caracteres a contener enteros, pues después operaremos con estos, así que se toman los valores de caracteres del 0 al 9 y son pasados a los valores 0 a 9 en versión entera, apoyándonos del código ascii y restando 48 espacios, en la línea 464.

Se crea una función que decodifica "decode", en la línea 32. Esta lo que hace es tomar las coordenadas, las cuales ahora vienen seguidas (se consideran el primer número en posición impar y el que le sigue, como un par de coordenadas) y las asocia a una posición específica de alguna lista de largo de fila igual a 75, como lo es el pad del don Quijote.

Esta función es usada en la lista con las coordenadas y se obtiene la lista con las posiciones en el Quijote, llamada "salida2_ decode". Esta lista es comparada espacio por espacio a la lista "pad_ tarea2" la cual contiene el Quijote, y de esta forma se asocia el valor de la posición a valores de caracteres, para lograr la decodificación en la lista de caracteres llamada "decodificado", en la línea 473. Finalmente se imprime la lista "decodificado"

en el texto de salida "salida_decod.txt", en la línea 479.

Observaciones:

El código tiene errores en la codificación, no son graves. Pero primero se observa que la función "azar" efectivamente toma valores al azar, pero son constantes dentro del texto.

Otro error que si es un poco grave es que a la hora de insertar las coordenadas para los valores "K" y "W" se tiene que imprime un par de coordenadas extras, que pueden ser ignoradas visualmente, pero a la hora de decodificar descuadra toda la lista. Una solución no implementada es simplemente eliminar aquellos elementos "pair" que estén antes de los pares (0,0) o (0,1), por ejemplo.

Otra observación ahora si grave, es que la sección de decodificación no es efectiva, aunque la lógica utilizada para desarrollarla tiene bastante sentido para los integrantes del grupo, tal vez un pequeño desfase en algún numero descuadra absolutamente toda la decodificación.