Trabajo Práctico Nº 1: Abstracción

Condiciones de entrega

El presente trabajo práctico deberá cumplir con todos los requisitos detallados en el reglamento de la cátedra. La fecha de entrega es el martes 8 de abril de 2008.

Introducción

Cifrar un mensaje quiere decir *transcribir el mismo en letras o símbolos, de acuerdo con una clave, con el fin de proteger su contenido.*

Dado un mensaje cifrado, **descifrarlo** quiere decir *obtener la representación original del mismo, valiéndose de la clave utilizada para cifrarlo.*

La clave es una *secuencia de letras o símbolos que controlan el funcionamiento del algoritmo de cifrado.* Si ciframos un mismo mensaje dos veces utilizando el mismo algoritmo, pero con claves distintas, entonces *es muy probable que el mensaje cifrado resultante sea distinto*. Además, para descifrar un mensaje *es necesario* conocer de antemano la clave utilizada para el cifrado.

En el ámbito de la informática, un **cifrador/descifrador** es un programa, módulo o biblioteca que permite cifrar y descifrar mensajes. Para este trabajo práctico, deberán desarrollar el **TDA Archivo Cifrado**, que permitirá leer y escribir líneas de texto en un archivo, encargándose de que las mismas se almacenen de forma cifrada para proteger su contenido. El algoritmo de cifrado a utilizar será una variante del conocido como **Playfair**, que llamaremos **Playfair II**.

Nota lingüística!

Es muy común escuchar a la gente decir el verbo "*encriptar"*, o alguno de sus derivados (como "*encriptador*", "*encriptación*", ...). De hecho muchas veces nosotros mismos caemos en su uso por simple repetición. La realidad es que el verbo "*encriptar*" no está aceptado por la *Real Academia Española de Letras* como una palabra del idioma español. Su uso es un anglicismo del término inglés "encrypt". Por supuesto, lo mismo sucede con su antónimo, "*desencriptar*".

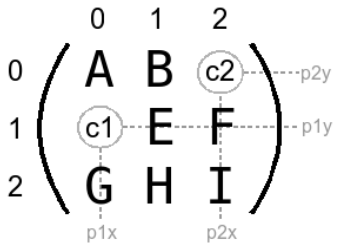
Playfair II

El primer paso para utilizar este algoritmo será definir el **juego de caracteres**, esto es, el *conjunto de todos caracteres y símbolos que podrán aparecer en los mensajes a cifrar*. Una vez definido un juego de caracteres de M elementos, debemos disponer los mismos arbitrariamente en una matriz cuadrada de NxN (donde $N^2 \geq M$), rellenando todos los lugares vacíos con caracteres que no formen parte del juego definido anteriormente. Esta matriz será la **clave** de nuestro sistema.

Algoritmo de cifrado

(1) Mientras queden caracteres para leer
  (a) Leer los dos siguientes caracteres en **c1** y **c2**.
  (b) Si **c1** = **c2**, escribir "&A", donde A es **c1** o **c2**.
  (c) Si **c1 ≠ c2**
    i. **(p1y, p1x)** := posición de **c1** en la matriz *(ver imagen)*.
    ii. **(p2y, p2x)** := posición de **c2** en la matriz.
    iii. Si **p1x = p2x** (pertenecen a la misma columna), escribir "$AB" donde A es el carácter en la posición **(p1y+1, p1x)** y B es el carácter en la posición **(p2y+1, p2y)**.
    iv. Si **p1y = p2y** (pertenecen a la misma fila), escribir "%AB" donde A es el carácter en la posición **(p1y, p1x+1)** y B es el carácter en la posición **(p2y, p2x+1)**.
    v. Si **p1x ≠ p2x** y **p1y ≠ p2y**, escribir "AB", donde A es el carácter en la posición **(p1y, p2x)** y B es el carácter en la posición **(p2y, p1x)**.

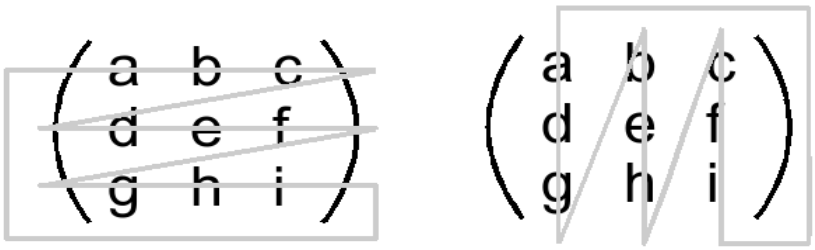
Algoritmo de descifrado



(1) Mientras queden caracteres para leer
  (a) Leer el siguiente carácter en **c**.
  (b) Si **c = "&"**
    i. Leer el siguiente carácter en **c1**.
    ii. Escribir **c1** dos veces.
  (c) Si **c = "$"**
    i. Leer los dos siguientes caracteres en **c1** y **c2**.
    ii. **(p1y, p1x)** := posición de **c1** en la matriz.
    iii. **(p2y, p2x)** := posición de **c2** en la matriz.
    iv. Escribir "AB" donde A es el carácter en la posición **(p1y-1, p1x)** y B es el carácter en la posición **(p2y-1, p2x)**.
  (d) Si **c = "%"**
    i. Leer los dos siguientes caracteres en **c1** y **c2**.
    ii. **(p1y, p1x)** := posición de **c1** en la matriz.
    iii. **(p2y, p2x)** := posición de **c2** en la matriz.
    iv. Escribir "AB" donde A es el carácter en la posición **(p1y, p1x-1)** y B es el carácter en la posición **(p2y, p2x-1)**.
  (e) Sino
    i. c1 := c
    ii. Leer el siguiente carácter en **c2**.
    iii. **(p1y, p1x)** := posición de **c1** en la matriz.
    iv. **(p2y, p2x)** := posición de **c2** en la matriz.
    v. Escribir "AB" donde A es el carácter en la posición **(p1y, p2x)** y B es el carácter en la posición **(p2y, p1x)**.

Observaciones

Los algoritmos tal cual descriptos presentan algunos casos patológicos. Por ejemplo, ¿qué sucede cuando se debe escribir el carácter en la posición **(p1y, p1x+1)** y **p1x** corresponde a la última columna de la matriz? La solución será escribir el carácter en la posición **(p1y+1, 0)**, esto es, el primer carácter de la siguiente fila. La figura que sigue muestra el criterio que debe seguirse para solucionar casos de este tipo.



Otro caso patológico se da cuando el mensaje a cifrar tiene una cantidad impar de letras, ya que nuestro algoritmo de cifrado nos exige leer pares de carácteres. La solución es bastante simple, y consiste en ocupar una posición de la matriz con el carácter "@". A la hora de cifrar un mensaje con cantidad impar de letras, entonces le agregaremos una *arroba* al final para que el mismo pase a tener una cantidad par de letras, y entonces podamos cifrarlo con el algoritmo expuesto anteriormente.

Un último caso patológico que debemos tratar es cuando el mensaje a cifrar contiene los caracteres especiales utilizados por nuestro algoritmo (i.e.: "&", "$", "%", "@"). En este caso aplicaremos la solucion trivial: prohibir estos símbolos en el juego de caracteres permitido.

Ejemplos

Cifrar la palabra "ALGORITMOS" utilizando la matriz dada

A	O	G
L	R	M
S	T	I

AL se cifra como **$LS**
GO se cifra como **%LG**
RI se cifra como **MT**
TM se cifra como **IR**
OS se cifra como **AT**

Por lo tanto, el mensaje cifrado resulta **$LS%LGMTIRAT**.

Cifrar la palabra "TRAGO" utilizando la matriz dada

Comenzamos agregando un signo @ al final del mensaje, ya que el mismo tiene una cantidad impar de letras. Por lo tanto, el mensaje a cifrar será **"TRAGO@".**

T	A	G
R	O	L
C	B	@

TR se cifra como $**RC**
AG se cifra como **%GR**
O@ se cifra como **LB**

Por lo tanto, el mensaje cifrado resulta $**RC%GRLB.**

Aclaración!
En la matriz del primer ejemplo no pusimos la arroba porque no era necesario para ese ejemplo en concreto, pero para que el algoritmo funcione en cualquier caso es necesario que la arroba aparezca en la matriz.

Descifrar el mensaje "%OD$AO&EEP$DR" utilizando la matriz dada

$$\begin{pmatrix} D & E & A \\ B & V & @ \\ P & O & R \end{pmatrix}$$

%OD se descifra como **PR**
$AO se descifra como **OV**
&E se descifra como **EE**
EP se descifra como **DO**
$DR se descifra como **R@**

Como el mensaje tiene un signo @ al final, entonces lo eliminamos para obtener el mensaje descifrado **PROVEEDOR.**

TDA Archivo Cifrado

Diseño del tipo de dato

La estructura que definirá al TDA Archivo Cifrado será la siguiente:

```
#define MAXBUFFER 255
#define N 9

typedef struct {
     char matriz[N][N]; // Clave del algoritmo Playfair
     FILE* archivo;
} TAC;
```

Primitivas

Deben desarrollarse las siguientes primitivas para el TDA, respetando exactamente la interfaz y los requerimientos indicados.

int TAC_Abrir (TAC* tac, char* archMensaje, char tipoAcceso, char* archConf)		
FUNCIÓN	Abre un archivo cifrado para lectura o escritura.	
PRE	`tac` tiene suficiente memoria reservada para una estructura de tipo `TAC`. `archMensaje` es una ruta de archivo válida. `tipoAcceso` es `r` o `w`. `archConf` es una ruta de archivo válida.	
POST	Si devuelve `0`, `tac` abierto. Si el tipo de acceso es de escritura y el archivo existe, se sobreescribe. Si hubo error al abrir el archivo de configuración devuelve `-1`. Si hubo error al abrir el archivo del mensaje, devuelve `-2`. Si el archivo de configuración es inválido devuelve `-3`.	
PARÁMETROS	`tac`	Manipulador del archivo cifrado.
	`archMensaje`	Ruta al archivo a abrir.
	`tipoAcceso`	`r` = lectura, `w` = escritura
	`archConf`	Ruta al archivo de configuración.

int TAC_Cerrar(TAC* tac)		
FUNCIÓN	Cierra el archivo cifrado `tac`.	
PRE	`tac` abierto.	
POST	Si devuelve `0`, `tac` cerrado. En otro caso devuelve `-1`.	
PARÁMETROS	`tac`	El manipulador del archivo a cerrar.

int TAC_Leer(TAC* tac, char[] mensaje)		
FUNCIÓN	Lee la siguiente línea del archivo cifrado.	
PRE	`tac` abierto. `mensaje` tiene al menos `MAXBUFFER` bytes reservados.	
POST	Si devuelve `0`, la siguiente línea del archivo se copió en `mensaje` (sin el carácter de fin de línea). Si devuelve `-1` se alcanzó el final del archivo. En cualquier otro caso devuelve `-2`.	
PARÁMETROS	`tac`	El manipulador del archivo desde el cual leer.
	`mensaje`	Buffer en el cual guardar la línea leída.

int TAC_Escribir(TAC* tda, char[] mensaje)		
FUNCIÓN	Escribe una línea en el archivo cifrado.	
PRE	`tac` abierto. `linea` es una cadena de caracteres de a lo sumo `MAXBUFFER` bytes.	
POST	Si devuelve `0`, escribió `mensaje` en una línea nueva al final del archivo. Si devuelve `-1` hubo error por caracteres inválidos en el mensaje. En cualquier otro caso devuelve `-2`.	
PARÁMETROS	`tac`	El manipulador del archivo al cual escribir.
	`mensaje`	Buffer con el mensaje a escribir.

Juego de caracteres

El juego de caracteres que se utilizará en este TP es el siguiente

- Las letras mayúscula y minúscula del alfabeto latino (52 caracteres).
- Las letras mayúscula y minúscula agregadas por el español (14 caracteres).
- Los dígitos del 0 al 9 (10 caracteres).
- Los símbolos punto **(.)** y coma **(,)**.
- El carácter de **espacio** y de **tabulación**.
- El carácter especial arroba **(@)**.

La matriz que se obtiene con este juego de caracteres es de 9x9.

Archivo de configuración

El archivo de configuración constará de nueve líneas de texto, con nueve caracteres cada una. El m-ésimo carácter de la n-ésima línea del archivo representa la posición (n, m) de la matriz (fila n, columna m).

Programa de aplicación

Se debe desarrollar un programa que, utilizando el **TDA Archivo Cifrado** definido más arriba, proponga las siguientes opciones al usuario:

- Cifrar un texto ingresado por consola y guardarlo en un archivo.

  En este caso el programa se invocará como:

  `tp1 <config> -c <cifrado>`

  `<config>` es la ruta al archivo de configuración, `<cifrado>` es la ruta al archivo donde se guardará el resultado. El programa se quedará esperando el ingreso de texto por parte del usuario. Una línea vacía indica el final del texto, en ese caso el programa deberá guardar el archivo y terminar.
- Abrir un archivo descifrado, cifrarlo y guardarlo en otro archivo.

  En este caso el programa se invocará como:

  `tp1 <config> -c <cifrado> <descifrado>`

  `<config>` es la ruta al archivo de configuración, `<cifrado>` es la ruta al archivo donde se guardará el resultado y `<descifrado>` es la ruta al archivo descifrado. El programa terminará luego de procesar el archivo.
- Abrir un archivo cifrado, descifrarlo y mostrarlo por consola.

  En este caso el programa se invocará como

  `tp1 <config> -d <cifrado>`

  `<config>` es la ruta al archivo de configuración y `<cifrado>` es la ruta al archivo cifrado. El programa terminará luego de mostrar el contenido del archivo.
- Abrir un archivo cifrado, descifrarlo y guardarlo en otro archivo.

  En este caso el programa se invocará como:

  `tp1 <config> -d <cifrado> <descifrado>`

  `<config>` es la ruta al archivo de configuración, `<cifrado>` es la ruta al archivo cifrado y `<descifrado>` es la ruta al archivo donde se guardará el resultado. El programa terminará luego de procesar el archivo.

El programa debe informar al usuario si la ejecución fue realizada con éxito o no. **Los mensajes de error deben ser claros y brindar la mayor cantidad de información posible al usuario.** Esto incluye, pero no se limita a, errores de sintaxis, errores originados en el manejo de archivos, y errores al invocar a las primitivas.