Universidad Central del Ecuador
Criptografía
Ing. Giovanny Moncayo

Integrantes:
Coloma Pozo Dillan Renato
Coyago Reinoso Henry Israel
Estrella Caicedo Juan Carlos
Llumiquinga Molina Daniel Bernardo
Ortega Salinas Jimmy Fabricio

Tarea:

1. Algoritmo que escriba **todas las permutaciones posibles de una palabra de longitud n SIN espacios (Anagrama)**. La palabra se ingresa al iniciar el algoritmo. El algoritmo debe mostrar el número total de permutaciones y las 10 primeras ordenadas alfabéticamente.

El código tiene como objetivo hacer el método de cifrado por permutación o conocido como cifrado por trasposición.

La criptografía es la ciencia que estudia los métodos y procedimientos, mediante algoritmos matemáticos, que permiten la confiabilidad, integridad y confidencialidad de la información. Triana Laverde, en su trabajo: aplicaciones matriciales a la criptografía, señala: “existen diversos métodos para encriptar o cifrar un mensaje” [1]

Un cifrado clásico es un canal para ocultar un mensaje, conocido como mensaje en claro, donde las letras son sustituidas o transpuestas por otras letras, pares de letras y algunas veces por muchas letras.[2]

En el ámbito de la criptografía, el cifrado clásico ha sido históricamente utilizado y en la actualidad, la mayoría de estos métodos se implementan mediante software informático. Los enfoques más contemporáneos emplean dispositivos informáticos u otras tecnologías digitales que operan a nivel de bits y bytes. Varios de estos cifrados clásicos fueron empleados por figuras prominentes como Julio César y Napoleón, quienes desarrollaron sus propias técnicas de cifrado, algunas de las cuales han ganado popularidad. Muchos de estos sistemas tienen un trasfondo militar. A veces, se incluyen junto a los cifrados clásicos otras máquinas mecánicas o electromecánicas, como es el caso de Enigma. [3]

Una versión temprana de un cifrado de transposición era un Scytale , en el que se envolvía papel alrededor de un palo y se escribía el mensaje. Una vez desenvuelto, el mensaje sería ilegible hasta que el mensaje se envolviera de nuevo alrededor de un palo del mismo tamaño.[4]

Un cifrado de transposición moderno se realiza escribiendo el mensaje en filas, luego formando el mensaje cifrado a partir del texto en las columnas.[4]

Cifre el mensaje “Meet at First and Pine at midnight” usando filas de 8 caracteres de largo.

Solución

Escribimos el mensaje en filas de 8 caracteres cada una. Los caracteres sin sentido se agregan al final para completar la última fila.

MEETATFI*MEETATFI*

RSTANDPI*RSTANDPI*

NEATMIDN*NEATMIDN*

IGHTPXNR*IGHTPXNR*

Transposición columnar simple

El cifrado con forma de columna. En él, el mensaje original estará limitado a un rectángulo, de izquierda a derecha y de arriba hacia abajo. Después, se escoge una clave para asignar un número a cada columna del rectángulo para determinar el orden. El número correspondiente a la letra de la clave estará determinado por orden alfabético. [7]

Código en JAVA:

```
package algoritmos;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;


public class AlgoritmoPermutacion {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Ingrese una palabra: ");
        String palabra = scanner.nextLine();

        ArrayList<String> permutaciones = permutarPalabra(palabra);
        Collections.sort(permutaciones);

        System.out.println("Número total de permutaciones: " +
permutaciones.size());
        System.out.println("Las 10 primeras permutaciones ordenadas
alfabéticamente:");

        for (int i = 0; i < 10 && i < permutaciones.size(); i++) {
            System.out.println(permutaciones.get(i));
        }
    }

    public static ArrayList<String> permutarPalabra(String palabra) {
        ArrayList<String> permutaciones = new ArrayList<>();
        permutarRecursivo("", palabra, permutaciones);
        return permutaciones;
    }
```

```
    private static void permutarRecursivo(String prefijo, String
palabra, ArrayList<String> permutaciones) {
        if (palabra.length() == 0) {
            permutaciones.add(prefijo);
        } else {
            for (int i = 0; i < palabra.length(); i++) {
                permutarRecursivo(prefijo + palabra.charAt(i),
palabra.substring(0, i) + palabra.substring(i + 1), permutaciones);
            }
        }
    }
}
```

Explicación del código Java:

1. Se importan las clases necesarias: `ArrayList`, `Collections` y `Scanner`.
2. Se define la clase `Permutaciones` con el método `main`.
3. En el método `main`, se solicita al usuario que ingrese una palabra.
4. Se llama al método `permutarPalabra` con la palabra ingresada y se almacenan todas las permutaciones en un `ArrayList`.
5. Se ordenan alfabéticamente las permutaciones utilizando `Collections.sort`.
6. Se imprime el número total de permutaciones y las 10 primeras permutaciones ordenadas alfabéticamente.
7. El método `permutarPalabra` es el encargado de generar todas las permutaciones recursivamente, utilizando el método auxiliar `permutarRecursivo`.
8. El método `permutarRecursivo` recibe tres parámetros: `prefijo` (la permutación actual), `palabra` (las letras restantes por permutar) y `permutaciones` (el `ArrayList` donde se almacenarán todas las permutaciones).
9. Si la longitud de `palabra` es cero, significa que se ha formado una permutación completa, por lo que se agrega el `prefijo` al `ArrayList` de `permutaciones`.
10. Si la longitud de `palabra` es mayor que cero, se itera sobre cada carácter de `palabra`, tomando cada carácter como el siguiente carácter de la permutación actual (`prefijo + palabra.charAt(i)`), y se realiza una llamada recursiva con el resto de caracteres (`palabra.substring(0, i) + palabra.substring(i + 1)`).

2. Algoritmo que realice el cifrado de un mensaje por **permutación de filas, teniendo como clave n filas**. Tanto n como el texto del mensaje se ingresan al iniciar el algoritmo. El algoritmo debe controlar que el número de caracteres del mensaje (sin espacios), sea menor o igual que n x n. Imprima la matriz de cifrado, el mensaje original y el mensaje cifrado. Si en la matriz de cifrado sobran espacios para almacenar los caracteres del mensaje original, estos deben llenarse con "*".

El Texto en claro se debe escribir en vertical, de arriba hacia abajo, en las columnas de una matriz de n filas por n columnas. Cada x filas, siendo x el número de la clave, se continua escribiendo el texto que falta en la columna de la derecha.[8]

Por esto el criptograma es el texto de la lectura de las filas de la matriz.[8]

- M = QUE LA FUERZA TE ACOMPAÑE
- Clave NF = 4, Relleno = X

Q	A	R	E	M	E
U	F	Z	A	P	X
E	U	A	C	A	X
L	E	T	O	Ñ	X

- C = QAREM EUFZA PXEUA CAXLE TOÑX

En criptografía, un cifrado por transposición es un tipo de cifrado en el que unidades de texto plano se cambian de posición siguiendo un esquema bien definido; las 'unidades de texto' pueden ser de una sola letra (el caso más común), pares de letras, tríos de letras, mezclas de lo anterior. Hay una permutación de 'unidades de texto'. Este tipo de métodos eran muy usados en la criptografía clásica y por tanto, al tener que hacer los cálculos por medios muy básicos, normalmente el algoritmo se basaba en un diseño geométrico o en el uso de artilugios mecánicos (Ej escítala). Este tipo de algoritmos son de clave simétrica porque es necesario que tanto el que cifra como el que descifra deben conocer la misma clave para realizar su función. La clave puede es intrínseca en el propio método de cifrado/descifrado de forma que algoritmo y clave son un conjunto indivisible.[5]

Entonces podríamos codificar el mensaje grabando las columnas. La primera columna, leyendo hacia abajo, sería MRNI. En conjunto, el mensaje codificado sería MRNI ESEG ETAH TATT ANMP TDIX FPDN IINR. Los espacios serían eliminados o reposicionados para ocultar el tamaño de la tabla utilizada, ya que esa es la clave de cifrado en este mensaje.[6]

Código en Java

```
package algoritmos;
import java.util.Scanner;
public class AlgoritmoPermutacionFilas {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Ingrese el número de filas (n): ");
        int n = scanner.nextInt();
        scanner.nextLine(); // Consumir el salto de línea restante

        System.out.print("Ingrese el mensaje: ");
        String mensaje = scanner.nextLine().replaceAll("\\s+", ""); //
Eliminar espacios en blanco
```

```
        if (mensaje.length() > n * n) {
            System.out.println("El mensaje es demasiado largo para la
matriz de cifrado.");
            return;
        }

        char[][] matrizCifrado = new char[n][n];

        // Llenar la matriz de cifrado con el mensaje de forma
recursiva
        llenarMatrizCifrado(matrizCifrado, mensaje, 0, 0);

        // Imprimir la matriz de cifrado
        System.out.println("Matriz de cifrado:");
        imprimirMatriz(matrizCifrado);

        // Imprimir el mensaje original
        System.out.println("Mensaje original: " + mensaje);

        // Imprimir el mensaje cifrado
        String mensajeCifrado = generarMensajeCifrado(matrizCifrado,
n);
        System.out.println("Mensaje cifrado: " + mensajeCifrado);
    }

    private static void llenarMatrizCifrado(char[][] matrizCifrado,
String mensaje, int fila, int columna) {
        if (columna == matrizCifrado[0].length) {
            fila++;
            columna = 0;
        }

        if (fila == matrizCifrado.length || mensaje.isEmpty()) {
            // Llenar los espacios restantes con '*'
            llenarConCaracter(matrizCifrado, fila, columna, '*');
            return;
        }

        matrizCifrado[fila][columna] = mensaje.charAt(0);
        llenarMatrizCifrado(matrizCifrado, mensaje.substring(1), fila,
columna + 1);
    }

    private static String generarMensajeCifrado(char[][]
matrizCifrado, int n) {
        StringBuilder mensajeCifrado = new StringBuilder();
        for (int columna = 0; columna < n; columna++) {
            for (int fila = 0; fila < n; fila++) {
                if (matrizCifrado[fila][columna] != '*') {

mensajeCifrado.append(matrizCifrado[fila][columna]);
                }
            }
        }
        return mensajeCifrado.toString();
    }

    private static void imprimirMatriz(char[][] matriz) {
        for (char[] fila : matriz) {
            for (char elemento : fila) {
```

```
                System.out.print(elemento + " ");
            }
            System.out.println();
        }
    }

    private static void llenarConCaracter(char[][] matriz, int
filaInicio, int columnaInicio, char caracter) {
        for (int fila = filaInicio; fila < matriz.length; fila++) {
            for (int columna = (fila == filaInicio) ? columnaInicio :
0; columna < matriz[0].length; columna++) {
                matriz[fila][columna] = caracter;
            }
        }
    }

}
```

Este código implementa un algoritmo de cifrado por permutación de filas. El cifrado se realiza mediante la creación de una matriz cuadrada de tamaño `n x n`, donde `n` es un número ingresado por el usuario. El mensaje para cifrar también es ingresado por el usuario y se elimina cualquier espacio en blanco.

El funcionamiento del código es el siguiente:

1. El programa solicita al usuario que ingrese el número de filas `n` y el mensaje a cifrar.
2. Se verifica si la longitud del mensaje (sin espacios) es mayor que `n x n`. Si es así, se muestra un mensaje de error y se finaliza el programa.
3. Se crea una matriz cuadrada de caracteres de tamaño `n x n`.
4. Se llama al método `llenarMatrizCifrado` de forma recursiva, que se encarga de llenar la matriz con los caracteres del mensaje. Si el mensaje se termina antes de llenar toda la matriz, los espacios restantes se llenan con el carácter `'*'`.
5. Se imprime la matriz de cifrado.
6. Se imprime el mensaje original (sin espacios).
7. Se llama al método `generarMensajeCifrado`, que recorre la matriz por columnas y genera el mensaje cifrado concatenando los caracteres distintos de `'*'`. El mensaje cifrado se imprime.

Que hace cada uno de los métodos que se implementó:

- `llenarMatrizCifrado`: Este método se encarga de llenar la matriz de cifrado con los caracteres del mensaje. Funciona de manera recursiva, recorriendo la matriz fila por fila y columna por columna. Si se alcanza el final de una fila, se pasa a la siguiente fila y se reinicia la columna. Si se alcanza el final de la matriz o el mensaje se termina, se llama al método `llenarConCaracter` para rellenar los espacios restantes con el carácter `'*'`. En caso contrario, se asigna el primer carácter del mensaje a la posición actual de la matriz y se llama recursivamente al método con el resto del mensaje y la siguiente posición de la matriz.
- `generarMensajeCifrado`: Este método genera el mensaje cifrado recorriendo la matriz por columnas. Utiliza dos bucles anidados para recorrer cada columna y

cada fila. Si el carácter en la posición actual de la matriz es distinto de `'*'`, se agrega al `StringBuilder` que construye el mensaje cifrado.

- `imprimirMatriz`: Este método imprime una matriz de caracteres. Utiliza un bucle anidado para recorrer todas las filas y columnas de la matriz, imprimiendo cada elemento separado por un espacio.
- `llenarConCaracter`: Este método llena una porción de la matriz con un carácter específico. Recibe la matriz, la fila y columna inicial, y el carácter con el que se rellenarán los espacios. Utiliza dos bucles anidados para recorrer las filas y columnas a partir de la posición inicial, asignando el carácter dado a cada posición.

3. Algoritmo que realice el cifrado de un mensaje por **permutación de columnas, teniendo como clave n columnas**. Tanto n como el texto del mensaje se ingresan al iniciar el algoritmo. El algoritmo debe controlar que el número de caracteres del mensaje (sin espacios), sea menor o igual que n x n. Imprima la matriz de cifrado, el mensaje original y el mensaje cifrado. Si en la matriz de cifrado sobran espacios para almacenar los caracteres del mensaje original, estos deben llenarse con "*".

```
class Ejercicio {

  public static void main(String[] args) {



      String mensaje = "holamundo";

      int n = 5;



      if (mensaje.length() > n * n) {

         System.out.println("El número de caracteres del mensaje (sin espacios) debe ser menor o igual que n x n.");

         return;

      }



      char[][] matriz = new char[n][n];

      int index = 0;



      // Llenar la matriz con el mensaje

      for (int i = 0; i < n; i++) {
```

```
    for (int j = 0; j < n; j++) {
        if (index < mensaje.length()) {
            matriz[i][j] = mensaje.charAt(index);
            index++;
        } else {
            matriz[i][j] = '*';
        }
    }
}

// Imprimir la matriz de cifrado
System.out.println("Matriz de cifrado:");
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        System.out.print(matriz[i][j] + " ");
    }
    System.out.println();
}

// Imprimir el mensaje original
System.out.println("Mensaje original: " + mensaje);

// Imprimir el mensaje cifrado
System.out.print("Mensaje cifrado: ");
for (int j = 0; j < n; j++) {
    for (int i = 0; i < n; i++) {
```

```
            System.out.print(matriz[i][j]);

        }

    }

  }


}
```

El funcionamiento del código es el siguiente:

1. Primero, se define un mensaje y un número n que determinará el tamaño de la matriz de cifrado (n x n).
2. Luego, se verifica si la longitud del mensaje es mayor que n * n. Si es así, se imprime un mensaje de error y se termina el programa.
3. Se crea una matriz de caracteres n x n y se llena con los caracteres del mensaje. Si el mensaje es más corto que n * n, los espacios restantes en la matriz se llenan con el carácter '*'.
4. Se imprime la matriz de cifrado, que muestra cómo se ha organizado el mensaje en la matriz.
5. Se imprime el mensaje original.
6. Finalmente, se imprime el mensaje cifrado. Para obtener el mensaje cifrado, se leen los caracteres de la matriz en el orden de las columnas, de arriba a abajo y de izquierda a derecha.

4. Algoritmo que realice el cifrado de una cadena de caracteres mediante un **método de sustitución Monoalfabético de desplazamiento n caracteres a la derecha**. Tanto la palabra como el valor de n se ingresan al iniciar el algoritmo. El algoritmo debe mostrar el alfabeto original, el alfabeto cifrado, la cadena de caracteres ingresada y su resultado.

Algoritmo en Python:

```
def cifrado_monoalfabetico(texto, desplazamiento):
   abecedario = 'abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTUVWXYZ'
   resultado = ''
   if desplazamiento > 0 :
    for caracter in texto:
       if caracter in abecedario:
          indice = (abecedario.index(caracter) + desplazamiento) % len(abecedario)
          resultado += abecedario[indice]
       else:
          resultado += caracter
    return resultado
   else:
    resultado = 'Error, Desplazamiento negativo, no se puede ejecutar'
    return resultado


palabra = input("Ingrese la palabra a cifrar: ")
desplazamiento = int(input("Ingrese el valor de desplazamiento: "))



if desplazamiento > 0:
 print("abecedario original:", 'abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTUVWXYZ')
 print("abecedario cifrado:",
cifrado_monoalfabetico('abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTUVWXYZ',
desplazamiento))
 print("Texto original:", palabra)
 print("Texto cifrado:", cifrado_monoalfabetico(palabra, desplazamiento))
else:
 print("No se puede ejecutar, el desplazamiento es negativo.")
```

Este algoritmo implementa un cifrado monoalfabético utilizando un desplazamiento fijo en el alfabeto, llamado también cifrado César. Es una de las técnicas de cifrado más simples y usadas. Es un tipo de cifrado por sustitución en el que una letra en el texto original se reemplaza por otra que está un número fijo de posiciones después en el alfabeto. [11]

Se define el alfabeto original en la variable abecedario, que incluye tanto letras minúsculas como mayúsculas, incluyendo la letra "ñ" en minúscula y mayúscula.

Luego se inicializa una cadena vacía llamada resultado donde se almacenará el texto cifrado.

Se itera sobre cada elemento en el texto de entrada.

Dentro del bucle, se comprueba si el elemento actual está presente en el alfabeto original. Si lo está:

- Se obtiene el índice del elemento en el alfabeto original usando abecedario.index(caracter).
- Se suma el desplazamiento al índice obtenido.
- Se utiliza el operador de módulo % para asegurarse de que el índice resultante esté dentro del rango del alfabeto.
- Se obtiene el caracter cifrado correspondiente al nuevo índice y se añade al resultado.
- Si el caracter no está en el alfabeto original (por ejemplo, un espacio o un carácter especial (*, >, °, etc.)), se añade directamente al resultado sin cifrarlo.

Cuando se cifró todo el texto, se devuelve el resultado.

5. Algoritmo que realice el cifrado de una cadena de caracteres mediante un **método de sustitución Polialfabético de Vigenère**. La cadena se ingresa al iniciar el algoritmo. El algoritmo debe mostrar la cadena de caracteres ingresada, la clave de cifrado y la cadena de caracteres cifrada.

El cifrado Vigenère es un tipo de cifrado de sustitución polialfabética utilizado para la encriptación de datos. Fue desarrollado por el criptógrafo francés Blaise de Vigenère en el siglo XVI, aunque la variante más importante fue introducida por Gilbert S. Vernam en 1918. Este cifrado se basa en la utilización de una palabra clave para desplazar cada letra del mensaje original en diferentes posiciones, lo que lo hace más resistente a los ataques criptoanalíticos. Sin embargo, su seguridad se vio comprometida por la necesidad de intercambiar una clave imprácticamente grande entre los comunicantes[9].

```
public class CifradoVigenere {

  // Método para cifrar un mensaje utilizando el cifrado de Vigenère

  public static String cifrarVigenere(String mensaje, String clave) {

    StringBuilder mensajeCifrado = new StringBuilder();

    String alfabeto = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    clave = clave.toUpperCase();

    int claveIndex = 0;

    // Iterar a través de cada carácter del mensaje
```

```
    for (char caracter : mensaje.toCharArray()) {
        if (Character.isLetter(caracter)) {
            char caracterUpper = Character.toUpperCase(caracter);
            int fila = alfabeto.indexOf(caracterUpper);
            int columna = alfabeto.indexOf(clave.charAt(claveIndex));
            // Cifrar el carácter utilizando el cifrado de Vigenère
            char caracterCifrado = alfabeto.charAt((fila + columna) % 26);
            mensajeCifrado.append(caracterCifrado);
            claveIndex = (claveIndex + 1) % clave.length();
        } else {mensajeCifrado.append(caracter);} }
    return mensajeCifrado.toString();}
// Método para descifrar un mensaje cifrado utilizando el cifrado de Vigenère
public static String descifrarVigenere(String mensajeCifrado, String clave) {
    StringBuilder mensajeDescifrado = new StringBuilder();
    String alfabeto = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    clave = clave.toUpperCase();
    int claveIndex = 0;
    // Iterar a través de cada carácter del mensaje cifrado
    for (char caracter : mensajeCifrado.toCharArray()) {
        if (Character.isLetter(caracter)) {
            char caracterUpper = Character.toUpperCase(caracter);
            int fila = alfabeto.indexOf(caracterUpper);
            int columna = alfabeto.indexOf(clave.charAt(claveIndex));
            // Descifrar el carácter utilizando el cifrado de Vigenère
            char caracterDescifrado = alfabeto.charAt((fila - columna + 26) % 26);
            mensajeDescifrado.append(caracterDescifrado);
```

```
            claveIndex = (claveIndex + 1) % clave.length();

        } else {            mensajeDescifrado.append(caracter);}

    }return mensajeDescifrado.toString(); }

  // Método principal que demuestra el uso del cifrado y descifrado de Vigenère

  public static void main(String[] args) {

    String mensaje = "CRIPTOGRAFIA"; String clave = "OCTAVO";

    // Cifrar el mensaje y mostrar el resultado

    String mensajeCifrado = cifrarVigenere(mensaje, clave);

    System.out.println("Mensaje cifrado: " + mensajeCifrado);

    // Descifrar el mensaje cifrado y mostrar el resultado

    String mensajeDescifrado = descifrarVigenere(mensajeCifrado, clave);

    System.out.println("Mensaje descifrado: " + mensajeDescifrado); }}
```

A continuación, se explica cómo funciona este proceso utilizando el código proporcionado:

Explicación

Preparación: Selecciona un mensaje que deseas cifrar y una clave secreta que será utilizada para el cifrado y el descifrado. Por ejemplo, consideremos el mensaje "CRIPTOGRAFIA" y la clave "OCTAVO".

1. Cifrado: Para cifrar el mensaje, se utiliza el método cifrarVigenere(mensaje, clave) de la clase CifradoVigenere. Este método recorre cada carácter del mensaje y lo reemplaza por otro carácter basado en la posición del carácter en el alfabeto y en la clave proporcionada.

   Paso 1: Convertimos el mensaje y la clave a mayúsculas para asegurar la consistencia.

   Paso 2: Iteramos a través de cada carácter del mensaje.

   Paso 3: Para cada carácter, calculamos su posición en el alfabeto y la posición correspondiente en la clave.

   Paso 4: Sumamos estas posiciones y obtenemos un nuevo carácter cifrado.

   Paso 5: Repetimos este proceso para cada carácter del mensaje y construimos el mensaje cifrado.

2. Descifrado: Para descifrar el mensaje cifrado, se utiliza el método descifrarVigenere(mensajeCifrado, clave). Este método realiza el proceso inverso al cifrado para recuperar el mensaje original.

Paso 1: Convertimos el mensaje cifrado y la clave a mayúsculas para asegurar la consistencia.

Paso 2: Iteramos a través de cada carácter del mensaje cifrado.

Paso 3: Para cada carácter, calculamos su posición en el alfabeto y la posición correspondiente en la clave.

Paso 4: Restamos estas posiciones y obtenemos un nuevo carácter descifrado.

Paso 5: Repetimos este proceso para cada carácter del mensaje cifrado y construimos el mensaje descifrado.

Resultado: Finalmente, imprimimos tanto el mensaje cifrado como el mensaje descifrado en la consola.

6. Algoritmo que realice el cifrado de una cadena de caracteres utilizando la siguiente **tabla de cifrado**:

*	A	S	D	F	G
Q	a	b	c	d	e
W	f	g	h	i	j
E	k	l	m	n	o
R	p	q	r	s	t
T	u	v	x	y	z

Se trata del cuadrado de Polibio, intentado en el siglo 2 a. C. y usado. [10] “Es un caso particular de cifrado de sustitución mono-alfabética” [10]. El cifrado de Polibio fue usa para poder enviar mensajes a través de las antorchas en la antigua Grecia [11]. Analizando podemos ver que se trata de una transformación, puedo simplificar el mensaje de 25 letras a un grupo de 10 letras.

La forma de usarlo es la siguiente, buscas la letra en el tablero y la reemplazas por la letra mayúscula de la izquierda y la letra mayúscula de arriba, ejemplo “a” es “QA”. Tiene un símil con la tabla de multiplicar.

Algoritmo

Clase: Literal6 (Main)

```
public class Literal6 {
    public static void main(String[] args) {

        Scanner scan=new Scanner(System.in);
        System.out.print("Ingrese la cadena a encriptar: ");
        String cadena = scan.nextLine();

        System.out.println();

        char[][] tablaCifrado = new char[6][6]; //Inicializo la tabla
de encriptado

        tablaCifrado[0][0]='*';
        tablaCifrado[0][1]='A';
        tablaCifrado[0][2]='S';
        tablaCifrado[0][3]='D';
        tablaCifrado[0][4]='F';
        tablaCifrado[0][5]='G';

        tablaCifrado[1][0]='Q';
        tablaCifrado[1][1]='a';
        tablaCifrado[1][2]='b';
        tablaCifrado[1][3]='c';
        tablaCifrado[1][4]='d';
        tablaCifrado[1][5]='e';

        tablaCifrado[2][0]='W';
        tablaCifrado[2][1]='f';
        tablaCifrado[2][2]='g';
        tablaCifrado[2][3]='h';
        tablaCifrado[2][4]='i';
        tablaCifrado[2][5]='j';

        tablaCifrado[3][0]='E';
        tablaCifrado[3][1]='k';
        tablaCifrado[3][2]='l';
        tablaCifrado[3][3]='m';
        tablaCifrado[3][4]='n';
        tablaCifrado[3][5]='o';

        tablaCifrado[4][0]='R';
        tablaCifrado[4][1]='p';
        tablaCifrado[4][2]='q';
        tablaCifrado[4][3]='r';
        tablaCifrado[4][4]='s';
        tablaCifrado[4][5]='t';

        tablaCifrado[5][0]='T';
        tablaCifrado[5][1]='u';
        tablaCifrado[5][2]='v';
        tablaCifrado[5][3]='x';
        tablaCifrado[5][4]='y';
        tablaCifrado[5][5]='z';


        Polibio cifrado=new Polibio(tablaCifrado);
```

```
        String cadenaCifrada=cifrado.cifrar(cadena); //llamado al
método


        System.out.println("Tabla de cifrado: ");
        for(int i = 0; i < 6; i++) { //Impresión de la tabla
            for(int j = 0; j < 6; j++) {
                System.out.print(cifrado.getTablaCifrado()[i][j] +
"  ");
            }
            System.out.println();
        }

        System.out.println("Mensaje Original: " + cadena);
        System.out.println("Mensaje Cifrado: " + cadenaCifrada);

    }
}
```

Clase: Polibio

```
public class Polibio {

    private final char[][] tablaCifrado;
    public Polibio(char[][] tablaCifrado){ //ingresa la tabla
        this.tablaCifrado=tablaCifrado;

    }


    public String cifrar(String cadena){
        StringBuilder cadenaCifrada = new StringBuilder();
        for (char c : cadena.toCharArray()) { //for each para recorrer
la cadena a encriptar
            boolean encontrado = false;
            for (int i = 0; i < tablaCifrado.length; i++) {
                for (int j = 0; j < tablaCifrado[i].length; j++) {
                    if (tablaCifrado[i][j] == c) {
                        //Entonces si el elemento de la tabla coincide
con el carácter,
                        // yo agrego al StringBuilder las coordenadas
de la tabla.
                        cadenaCifrada.append(tablaCifrado[i][0]).appen
d(tablaCifrado[0][j]);
                        encontrado = true;
                        break;
                    }
                }
                if (encontrado) {
                    break;
                }
            }
            if (!encontrado) {
                //si no se llega a encontrar agrego **
                cadenaCifrada.append("**");
            }
        }
        return cadenaCifrada.toString();
    }

    public char[][] getTablaCifrado() {
        return tablaCifrado;
```

```
        }
}
```

Explicación

Se crea e ingresa la matriz que vamos a usar al constructor de la clase Polibio, luego por medio del objeto debemos ingresar los caracteres.

Dentro del método cifrar se usa un algoritmo elemental de recorrido de matrices, donde hacemos un triple for, donde recorremos cada letra de la cadena de caracteres y luego cada fila y columna de la tabla, para finalmente encontrar sus respectivos reemplazos.

Bibliografía

[1] J. Triana Laverde, *Aplicaciones matriciales a criptografía*. 2011.

[2] G. Della Porta, *"Sobre las notas secretas de las letras, comúnmente llamadas cifras, en 4 libros*, Mariam Scotum. Londres, 1591.

[3] G. Granados. Paredes, "Introducción a la criptografía", 2006.

[4] LibreTexts Español, "Cifrados de transposición".

[5] Prof. Ricardo Barraza y Prof. David Segura, "Criptografía Métodos de Sustitución y transposición." Consultado: el 29 de abril de 2024. [En línea]. Disponible en: https://educacion.sanjuan.edu.ar/mesj/LinkClick.aspx?fileticket=iXTmKMqhILs%3D&tabid=678&mid=1743

[6] LibreTexts Español, "Cifrados de transposición".

[7] Wikiwand, "Cifrado por transposición". Consultado: el 29 de abril de 2024. [En línea]. Disponible en: https://www.wikiwand.com/es/Cifrado_por_transposici%C3%B3n

[8] Jorge Aguirre, "Cripografía para ingenieros".

[9] "Vernam-Vigenere cipher | Definition, Advantages, Disadvantages, & Facts | Britannica". Consultado: el 29 de abril de 2024. [En línea]. Disponible en: https://www.britannica.com/topic/Vernam-Vigenere-cipher

[10] "Cuadrado de Polibio - Wikipedia, la enciclopedia libre". Consultado: el 29 de abril de 2024. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Cuadrado_de_Polibio

[11] "Polibio | Criptografía Clásica". Consultado: el 29 de abril de 2024. [En línea]. Disponible en: https://joseluistabaracarbajo.gitbooks.io/criptografia-clasica/content/Cripto04.html