MANUAL TÉCNICO

Carnet: 201904013

MANUAL TÉCNICO

MÉTODO PARA GUARDAR EL TEXTO INGRESADO YA SEPARADO EN CARACTERES.

```
static void GUARDAR_EN_VECTOR() {

    SEPARAR = new char[tamaño];
    ASCII = new int[tamaño];

    for (int posicion = 0; posicion < tamaño; posicion++) {
        //Ciclo para guardar caracteres
        char VALOR;

        //Se guarda el caracter
        VALOR = TXTCIFRAR.charAt(posicion);
        int ca = (int) (VALOR);
        //Se guarda en el vector
        SEPARAR[posicion] = VALOR;
        ASCII[posicion] = ca;

        //System.out.print(SEPARAR[posicion]+ ",");
        //System.out.print(ASCII[posicion] + ",");
    }
    //System.out.println("");
    //System.out.println("el tamaño del arreglo es:" + tamaño);
}
```

En este método se utilizan dos arreglos unidimendionales, uno de tipo char para separar por caracteres y uno tipo int para castear implícitamente el el arreglo de tipo char posición por posición y guardarlo.

MÉTODO PARA LLENAR UNA MATRIZ

```
static void LLENARMATRIZ() {
    /*METODO PARA LLENAR LA MATRIZ*/
    Matriz = new int[FILAS][COLUMNAS];

    int c = 0;
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            /*se da la posicion del valor*/
            Matriz[i][j] = ASCII[c];
            c++;
        }
    }
}
```

En este método se crea un arreglo bidimensional con el tamaño filas* columnas, se crean dos for anidados para ir dando una posición [i][j] en la matriz y guardar el valor del arreglo en esa posición.

MÉTODO PARA RECORRER O IMPRIMIR UNA MATRIZ

```
static void IMPRIMIR_MATRIZ2() {
    for (int i = 0; i < FILAS2; i++) {
        for (int j = 0; j < COLUMNAS2; j++) {
            /*Se imprime la posiciond de la matriz*/
            //System.out.print("|" + Matriz2[i][j] + "|");

        }
       // System.out.println("");
    }
}
```

Este método sirve para recorrer una matriz y verificar los valores que tiene en cada posición para posteriormente mostrarlos en consola.

MÉTODO PARA VERIFICAR EL LARGO DE UN TEXTO

```
static void VerificarLARGO() {

    if (tamaño % 3 == 0) {
        FILAS = 3;
        COLUMNAS = tamaño / 3;
        //System.out.println("El numero " + tamaño + "es multiplo de 3");
        //System.out.println("Matriz de" + FILAS + "x" + COLUMNAS);

    } else if (tamaño % 4 == 0) {

        FILAS = 4;
        COLUMNAS = tamaño / 4;
        //System.out.println("El numero " + tamaño + "es multiplo de 4");
        //System.out.println("Matriz de" + FILAS + "x" + COLUMNAS);

    } else if (tamaño % 5 == 0) {
```

En este método se verifica el tamaño del texto ingresado y se clasifica según su múltiplo en un rango de 3 a 17.

MÉTODO PARA LEER UN ARCHIVO

```
static void LEERARCHIVO() {
    try {
        String ruta;
        System.out.println("Ingrese la ruta del archivo");
        ruta = entrada.nextLine();
        File archivo = new File(ruta);
        FileReader fr = new FileReader(archivo);
        BufferedReader br = new BufferedReader(fr);

        //LEER ARCHIVO
        int[][] matriz;
        FILAS2 = 0;
        COLUMNAS2 = 0;
        String linea;
        String temporal = "";

        while ((linea = br.readLine()) != null) {
            String[] fila = linea.split(",");

            temporal += linea + ",";

            for (int i = 0; i < fila.length; i++) {

            }
            COLUMNAS2 = fila.length;
            FILAS2 = FILAS2 + 1;
```

En este método se lee la ruta de archivo con el filereader, se interpreta línea por línea y se guarda en un arreglo.

```
        matriz = new int[FILAS2][COLUMNAS2];
        //System.out.println("");
        //System.out.println("Numero de filas=" + FILAS2);
        //System.out.println("Numero de columnas=" + COLUMNAS2);

        String[] DATOS = temporal.split(",");
        TAM = DATOS.length;
        INTCADENA = new int[TAM];
        for (int c = 0; c < DATOS.length; c++) {
            int convertido = Integer.parseInt(DATOS[c]);
            INTCADENA[c] = convertido;
        }
        LLENARMATRIZ2();
        IMPRIMIR_MATRIZ2();
    } catch (Exception e) {
        e.printStackTrace();
    }
```

Se crea una matriz y un arreglo de tipo string para guardar lo leído para posteriormente guardarlo en un arreglo de tipo int.

MÉTODO PARA MULTIPLICAR MATRICES

```
static void MULTIPLICARMATRICES() {
    //System.out.println("MULTIPLICANDO MATRICES");
    //LLENARMATRIZ2();
    //IMPRIMIR_MATRIZ2();

    //System.out.println("x ");

    //LLENARMATRIZ();
    //IMPRIMIR_MATRIZ();
    System.out.println("********************");
    System.out.println("El mensaje cifrado es :");
    System.out.println("********************");
    CIFRADA = new int[FILAS2][COLUMNAS];

    for (int i = 0; i < FILAS2; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            for (int h = 0; h < COLUMNAS2; h++) {
                CIFRADA[i][j] += Matriz2[i][h] * Matriz[h][j];
            }
            System.out.print(CIFRADA[i][j] + ",");
        }
    }
}
```

En este método se define el tamaño de la matriz cifrada para posteriormente recorrerla y guardar en ella el resultado del producto de las otras dos matrices.

MÉTODO PARA CALCULAR EL DETERMINANTE DE UNA MATRIZ

```java
public double DETERMINANTE(int i, double[][] matriz) {
    if (matriz.length == 2) {
        /*si la matriz es 2x2*/
        double Deter = matriz[0][0] * matriz[1][1] - matriz[0][1] * matriz[1][0];

        return Deter;
        /*devuelve el determinante*/
    } else {
        double Deter = 0;

        for (int j = 0; j < matriz.length; j++) {
            /*se elimina la primera fila y columna de la matriz*/
            double[][] temp = this.SubMatriz(i, j, matriz);
            Deter = Deter + Math.pow(-1, i + j) * matriz[i][j] * this.DETERMINANTE(i, temp);
        }
        return Deter;
    }

}
```

Este método recibe una variable int que es la posición inicial y la matriz a la que se le calcula el determinante, en el ciclo for se anula la columna 1 y fila 1 diciendo this.Submatriz para finalmente retornar el determinante.

MÉTODO PARA CALCULAR LA SUBMATRIZ

```java
private double[][] SubMatriz(int i, int j, double[][] matriz) {
    /*se crea la submatriz*/
    double[][] temp = new double[matriz.length - 1][matriz.length - 1];
    int count1 = 0;
    int count2 = 0;
    for (int k = 0; k < matriz.length; k++) {
        if (k != i) {
            count2 = 0;
            for (int l = 0; l < matriz.length; l++) {
                if (l != j) {
                    temp[count1][count2] = matriz[k][l];
                    count2++;
                }

            }
            count1++;

        }

    }
    return temp;/*Retorna la matriz temporal creada*/
}
```

En este método se crea la submatriz anulando la fila 1 y columna uno de la matriz ingresada y retorna la matriz temporal creada.

MÉTODO PARA CALCULAR LA TRANSPUESTA DE UNA MATRIZ

```java
public double[][] Transpuesta(double[][] matriz) {

    double[][] tempTrans = new double[matriz.length][matriz.length];

    for (int i = 0; i < tempTrans.length; i++) {
        for (int j = 0; j < tempTrans.length; j++) {

            /*aqui se tranpone la matriz*/

        tempTrans[i][j]=matriz[j][i];
        }
    }
    return tempTrans;
}
```

Este método pide una matriz de tipo double y se le asigna el mismo tamaño de la matriz a trasponer, en los for anidados se transpone la matriz y se guarda en una matriz temporal la cual retorna el método.

DIAGRAMA DE FLUJO GENERAL