Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Laboratorio Introducción a la Programación y Computación 1

Manual Técnico

Para Practica 1:

Sistema de Cifrado, Descifrado y resolución de matrices por Gauss-Jordan

Hecho por:

Walter Daniel Jiménez Hernández 201901108

Fecha: 23/08/2020

Contenido

Introducción

La práctica 1 contiene 3 funcionalidades específicas, la primera es cifrar mensajes de cualquier tamaño, la segunda es descifrar mensajes que han sido cifrados utilizando la técnica anterior y la tercera es resolver sistemas lineales de 3 incógnitas por el método de Gauss-Jordan. Este manual tiene incluidos todos los aspectos técnicos que serán necesarios saber para poder manejar este programa.

Objetivos

El objeto de este manual es orientar y guiar a las personas técnicas en todos los aspectos que componen el programa de la práctica 1.

- Reconocer los métodos utilizados para cada una de las opciones.
- Guía para poder manejar todos los recursos del programa.

Este sistema va dirigido para las personas que desean cifrar mensajes de una manera rápida y segura, además solo con la clave se pueden descifrar los mensajes. También va dirigido para las personas que desean resolver sistemas lineales de 3 incógnitas de una manera rápida.

Conocimientos previos

- Programación Java (Básico)
- Programación orienta a objetos (Básico)
- HTML (Básico)
- Css (Básico)

Especificaciones Técnicas

Requisitos de Hardware

- Procesador Pentium 2 o superior.
- 256 MB de RAM (Mínimo)

Requisitos de Software

- **Sistema operativo**
  - Windows Vista
  - Windows7
  - Windows 8
  - Windows 10
  - Ubuntu
  - Debian

- **Lenguaje de Programación**
  - Java 8

- **Tecnologías Usadas**
  - IDE NetBeans 8.2
  - IntelliJ IDEA EDU

Lógica del programa

Clases Utilizadas

- **Constructor**

```
public Menu(){
    Scanner leer = new Scanner(System.in);
    byte opcion = 0;
    while (opcion!=4) {
        escribir("***************************** Menú *****************************\n"
                + "*                                                              *\n"
                + "* 1) Cifrar mensaje                                            *\n"
                + "* 2) Descifrar mensaje                                         *\n"
                + "* 3) Resolver matriz utilizando el método de Gauss-Jordan      *\n"
                + "* 4) Salir                                                     *\n"
                + "*                                                              *\n"
                + "****************************************************************");


        opcion = leer.nextByte();
        switch (opcion) {
            case 1:
                Cifrar();
                break;
            case 2:
                Descifrar();
                break;
            case 3:
                Gauss();
                break;
            case 4:
                escribir("Programa finalizado");
                break;
            default:
                escribir("Opción no valida");
```

- **Funciones**
    - **Escribir:** Almacena el System.out.println(“mensaje”), es para tener una palabra mas accesible cada vez que se va a mostrar algo en consola.

```
private void escribir(String mensaje){
    System.out.println(mensaje);
}
private void escribirsinsalto(String mensaje){ System.out.print(mensaje);}
```

    - **Cifrar:** Muestra el menú de la opción Cifrar.

```
private void Cifrar(){

    Scanner leer = new Scanner(System.in);

    try{
        escribir("************************* Cifrado **************************\n\n"
                + "Ingrese el mensaje que desea cifrar: (Mínimo 3 carácteres)");
        String texto = leer.nextLine();
        reporte="<h1>El texto ingresado es: </h1>\n" +
                "<p>"+texto+"</p>";
        int matrizascii[][] = convertirascii(texto);
        escribir("Ingrese la ruta del archivo en donde se ecuentra la matriz:");
        String ruta = leer.nextLine();
        int matrizentrada[][] = leermatriz(ruta,texto);
        escribir("");
        escribir("El mensaje \""+ texto+ "\" cifrado es:");
        escribir("");
        cifrado(matrizascii,matrizentrada,filas(texto));
        escribir("");
        generarreporte("Cifrado","ReporteCifrado");
    }catch(Exception e){
        escribir("Ocurrio un error: "+ e.getMessage());
    }
}
```

- **Convertirascii:** Convierte un texto en una matriz de N*M y también genera una matriz con los valores ASCII de la matriz de texto.

```
private int[][] convertirascii(String texto){
    try{
        int largo = texto.length();
        int numfilas = filas(texto);
        int numcolumnas = 0;
        if(numfilas>=3){
            numcolumnas = largo/numfilas;
            int i = 0;
            int j = 0;
            int matrizi[][] = new int[numfilas][numcolumnas];
            char matrizs[][]= new char[numfilas][numcolumnas];

            for (char caracter : texto.toCharArray()) {
                matrizs[i][j] = caracter;
                matrizi[i][j] = caracter;
                j++;
                if(j%numcolumnas==0){
                    j=0;
                    i++;
                }
            }
```

```
            generarmatrizchar("Matriz de texto",matrizs);
            generarmatrizint("Matriz del texto convertido a ASCII",matrizi);
            return matrizi;

        }else {
            escribir("El mensaje debe tener más de 3 carácteres");
            return null;
        }
    }catch (Exception e){
        escribir("Ocurrio un error " + e.getMessage());
        return null;
    }
}
```

- **Filas:** Calcula el número de filas de la matriz de texto.

```
private int filas(String texto){
    int largo = texto.length();
    int numfilas = 0;
    if(largo>=3) {
        if (largo % 3 == 0) {
            numfilas = 3;
        } else if (largo % 4 == 0) {
            numfilas = 4;
        } else if (largo % 5 == 0) {
            numfilas = 5;
        } else if (largo % 7 == 0) {
            numfilas = 7;
        } else if (largo % 11 == 0) {
            numfilas = 11;
        } else if (largo % 13 == 0) {
            numfilas = 13;
        } else if (largo % 17 == 0) {
            numfilas = 17;
        } else {
            escribir("Solo se tiene contemplado hasta multiplo de 17");
        }
        return numfilas;
    }else{
        return 0;
    }
}
```

- **Leermatriz:** Lee una matriz de enteros.

```
private int[][] leermatriz(String ruta, String texto){
    try{
        FileReader archivo = new FileReader("./"+ruta);
        Scanner leerarchivo = new Scanner(archivo);
        int longitud = filas(texto);
        int matriz[][] = new int[longitud][longitud];
        int j=0;
        while(leerarchivo.hasNextLine()){
            String linea[] = leerarchivo.nextLine().split(",");
            for(int i = 0; i<linea.length;i++){
                matriz[j][i]=Integer.parseInt(linea[i]);
            }
            j++;
        }
        generarmatrizint("Matriz clave",matriz);
        return matriz;

    }catch(FileNotFoundException e){
        escribir("No se encuentra el archivo");
        e.printStackTrace();
        return null;
    }
}
```

- **Cifrado:** Multiplica la matriz de enteros ASCII con la matriz clave.

```
private void cifrado(int matrizascii[][], int matrizentrada[][], int filas){
    int matrizcifrada[][] = new int[matrizascii.length][matrizascii[0].length];
    for(int i=0;i<filas;i++){
        for(int j=0;j<matrizascii[i].length;j++) {
            for (int k = 0; k < filas; k++) {
                matrizcifrada[i][j] += (matrizentrada[i][k] * matrizascii[k][j]);
            }
            escribirsinsalto(String.valueOf(matrizcifrada[i][j])+" \t");
        }
        escribir("");
    }
    escribir("");
    generarmatrizint("Matriz del resultante",matrizcifrada);
}
```

- **Descifrar:** Muestra el menú de la opción descifrar.

```
private void Descifrar(){
    Scanner leer = new Scanner(System.in);

    try{
        BigDecimal decimal;
        escribir("************************* Descifrado *************************\n\n"
                + "Ingrese la ruta de la matriz clave (N*N)");
```

```
    String ruta = leer.nextLine();
    calculofilas1(ruta);
    int matrizclave[][] = leermatrizclave(ruta);
    escribir("Ingrese la ruta de matriz del texto cifrado (N*M)");
    ruta = leer.nextLine();
    calculocolumnas1(ruta);
    int fraseint[][] = leermatrizascii(ruta);
    double matrizclaveinversa[][] = inversa(matrizclave);
    double matrizdescifrada[][] = new double[matrizclaveinversa.length][fraseint[0].length];

    for(int i=0;i<matrizclaveinversa.length;i++){
        for(int j=0;j<fraseint[i].length;j++) {
            for (int k = 0; k < matrizclaveinversa.length; k++) {
                matrizdescifrada[i][j] += (matrizclaveinversa[i][k] * fraseint[k][j]);
            }
        }
    }
    generarmatrizdb("Se multiplica la inversa de la clave por la matriz de texto cifrado", matrizdescifrada);

    for (int i=0;i< matrizdescifrada.length;i++){
        for(int j=0;j< matrizdescifrada[0].length;j++){
            decimal = new BigDecimal(matrizdescifrada[i][j]);
            matrizdescifrada[i][j] = decimal.setScale(0, BigDecimal.ROUND_HALF_UP).intValue();
        }
    }

        generarmatrizdb("Se aproxima al entero más cercano",matrizdescifrada);
        char matrizdescifradachar[][]=new char[matrizdescifrada.length][matrizdescifrada[0].length];
        String mensajedescifrado="";
        for (int i=0;i< matrizdescifrada.length;i++){
            for(int j=0;j< matrizdescifrada[0].length;j++){
                matrizdescifradachar[i][j] = (char)((int)matrizdescifrada[i][j]);
                mensajedescifrado += matrizdescifradachar[i][j];
            }
        }
        generarmatrizchar("Se convierte a char la matriz descifrada",matrizdescifradachar);

        escribir("El mesaje es: " + mensajedescifrado);
        reporte += "<h2>El mensaje descifrado es:" + mensajedescifrado+"<h2>";


        generarreporte("Descifrar","ReporteDescifrar");
    }catch(Exception e){
        escribir("Ocurrio un error: "+ e.getMessage());
    }
}
```

- **Calculocolumnas1:** Calcula el número de columnas que tiene la matriz ingresada para descifrar.

```
private void calculocolumnas1(String ruta) {
    try{
        Scanner leerint = new Scanner(new FileReader(ruta));
        String fila[]= leerint.nextLine().split(",");
        columnasde= fila.length;
    }catch(FileNotFoundException e){
        escribir("No se encuentra el archivo");
        e.printStackTrace();
        columnasde=0;
    }
}
```

- **Calculofilas1:** Calcula el número de filas que tiene la matriz ingresada para descifrar.

```
private void calculofilas1(String ruta) {
    try{
        Scanner leerint = new Scanner(new FileReader(ruta));
        String fila[]= leerint.nextLine().split(",");
        filasde= fila.length;
    }catch(FileNotFoundException e){
        escribir("No se encuentra el archivo");
        e.printStackTrace();
        filasde=0;
    }
}
```

- **Leermatrizclave:** Lee la matriz clave ingresada por el usuario.

```
private int[][] leermatrizclave(String ruta) {
    try{
        Scanner leerint = new Scanner(new FileReader(ruta));
        int matriz[][] = new int[filasde][filasde];
        int i = 0;
        while(leerint.hasNextLine()){
            String linea[] = leerint.nextLine().split(",");
            for (int j=0;j< linea.length;j++){
                matriz[i][j]= Integer.parseInt(linea[j]);
                escribirsinsalto(matriz[i][j]+"\t");
            }
            escribir("");
            i++;
        }
        generarmatrizint("Matriz clave ingresada",matriz);
        return matriz;
    }catch(FileNotFoundException e){
        escribir("No se encuentra el archivo");
        e.printStackTrace();
        return null;
    }
}
```

- **Leermatrizascii:** Lee la matriz con el texto cifrado.

```
private int[][] leermatrizascii(String ruta) {
    try{
        Scanner leerint = new Scanner(new FileReader(ruta));
        int matriz[][] = new int[filasde][columnasde];
        int i = 0;
        while(leerint.hasNextLine()){
            String linea[] = leerint.nextLine().split(",");
            for (int j=0;j< linea.length;j++){
                matriz[i][j]= Integer.parseInt(linea[j]);
                escribirsinsalto(matriz[i][j]+" \t");
            }
            escribir("");
            i++;
        }
        generarmatrizint("Matriz con texto cifrado ingresada",matriz);
        return matriz;
    }catch(FileNotFoundException e){
        escribir("No se encuentra el archivo");
        e.printStackTrace();
        return null;
    }
}
```

- **Inversa:** Hace el cálculo de la matriz inversa para la matriz clave.

```
private double[][] inversa(int[][] matrizclave) {

    double matrizinversa[][]=new double[matrizclave.length][matrizclave[0].length];
    for(int i=0;i< matrizinversa.length;i++){
        for (int j=0;j<matrizinversa.length;j++){
            matrizinversa[i][j]=matrizclave[i][j];
        }
    }
    int filasin = matrizinversa.length;
    double sustituir[][] = new double[filasin][filasin];
    double b[][] = new double[filasin][filasin];
    int indice[] = new int[filasin];
    for (int i=0; i<filasin; ++i) {
        b[i][i] = 1;
    }

    generarmatrizdb("Matriz triangula para operar", b);

    gausssuperior(matrizinversa,indice);

    for (int i=0; i<filasin-1; ++i){
        for (int j=i+1; j<filasin; ++j){
            for (int k=0; k<filasin; ++k){
                b[indice[j]][k]-= matrizinversa[indice[j]][i]*b[indice[i]][k];
            }
        }
    }
```

```
    generarmatrizdb("Al valor de la matriz b se le restan la matriz inversa por la matriz b",b);

    for (int i=0; i<filasin; ++i){
        sustituir[filasin-1][i] = b[indice[filasin-1]][i]/matrizinversa[indice[filasin-1]][filasin-1];
        for (int j=filasin-2; j>=0; --j){
            sustituir[j][i] = b[indice[j]][i];
            for (int k=j+1; k<filasin; ++k){
                sustituir[j][i] -= matrizinversa[indice[j]][k]*sustituir[k][i];
            }
            sustituir[j][i] /= matrizinversa[indice[j]][j];
        }
    }
    generarmatrizdb("Se sustituye de atras hacia adelante en la matriz invera", sustituir);
    return sustituir;
}
```

- **Gausssuperior:** Utiliza el método de gauss para realizar eliminaciones en el cálculo de la matriz inversa.

```
private void gausssuperior(double[][] matrizinversa, int[] indice) {
    int indicel = indice.length;
    double c[] = new double[indicel];

    for (int i=0; i<indicel; ++i){
        indice[i] = i;
    }
    for (int i=0; i<indicel; ++i){
        double c1 = 0;
        for (int j = 0; j < indicel; ++j){
            double c0 = Math.abs(matrizinversa[i][j]);
            if (c0 > c1){
                c1 = c0;
            }
        }
        c[i] = c1;
    }
    int k = 0;
    for (int j=0; j<indicel-1; ++j){
        double pi1 = 0;
        for (int i = j; i < indicel; ++i){

            double pi0 = Math.abs(matrizinversa[indice[i]][j]);
            pi0 /= c[indice[i]];
            if (pi0 > pi1) {
                pi1 = pi0;
                k = i;
            }
        }
```

```
            int itmp = indice[j];
            indice[j] = indice[k];
            indice[k] = itmp;
            for (int i = j + 1; i < indice1; ++i) {
                double pj = matrizinversa[indice[i]][j] / matrizinversa[indice[j]][j];

                matrizinversa[indice[i]][j] = pj;

                for (int l = j + 1; l < indice1; ++l) {
                    matrizinversa[indice[i]][l] -= pj * matrizinversa[indice[j]][l];
                }
            }
        }
        generarmatrizdb("Se utiliza el método de Gauss para hacer eliminación", matrizinversa);
    }
```

- **Gauss:** Es el menú de la opción de resolver matriz por el método de Gauss-Jordan.

```
private void Gauss(){
    Scanner leer = new Scanner(System.in);
    try{
        escribir("******************** Matriz Gauss-Jordan ********************\n");
        escribir("Ingrese la ruta del archivo en donde se ecuentra la matriz:");
        String ruta = leer.nextLine();
        escribir("La matriz ingresada es:");
        int matrizentrada[][] = leermatrizGauss(ruta);
        if(matrizentrada.length==3 && matrizentrada[0].length==4){
            resolverGauss(matrizentrada);
            generarreporte("Gauss Jordan","ReporteGauss");
        }else{
            escribir("La matriz debe ser de 3*4 Para poder resolverla");
        }
    }catch(Exception e){
        escribir("Ocurrio un error: "+ e.getMessage());
    }
}
```

- **ResolverGauss:** Hace todas las operaciones para poder las matrices.

```
    private void resolverGauss(int[][] matrizentrada) {
        double matriz[][]=new double[matrizentrada.length][matrizentrada[0].length];

        if(matrizentrada.length == 3 && matrizentrada[0].length == 4){
            if(matrizentrada[0][0]!=1){
            for(int i=0;i< matrizentrada[0].length;i++){
                matriz[0][i]=(matrizentrada[0][i]/matrizentrada[0][0]);
            }
            generarmatrizdb("Se vuelve 1 la posición [1][1], si es necesario",matriz);
        }else{
            for(int i=0;i< matrizentrada[0].length;i++){
                matriz[0][i]=matrizentrada[0][i];
            }
        }
        for(int i=1;i< matrizentrada.length;i++){
            for(int j=0;j<matrizentrada[0].length;j++){
                matriz[i][j]=matrizentrada[i][j]-(matriz[0][j]*matrizentrada[i][0]);
            }
        }
```

```
    generarmatrizdb("Se vuelve 0 la fila 2 y 3 en la columna 1",matriz);
    double indice2= matriz[1][1];
    for(int i=1;i< matrizentrada[0].length;i++){
        matriz[1][i]=matriz[1][i]/indice2;
    }
    generarmatrizdb("Se vuleve 1 la posición [2][2]",matriz);
    double valor1 = matriz[0][1];
    double valor3 = matriz[2][1];

    for(int i=0;i< matrizentrada.length;i++){
        for(int j=1;j<matrizentrada[0].length;j++){
            if (i==1){
                continue;
            }else if(i==0){
                matriz[i][j]=matriz[i][j]-(matriz[1][j]*valor1);
            }else{
                matriz[i][j]=matriz[i][j]-(matriz[1][j]*valor3);
            }
        }
    }
    generarmatrizdb("Se vuelve 0 el resto de filas en la columna 2", matriz);
    double indice3 = matriz[2][2];
    for(int j=2;j<matrizentrada[0].length;j++){
        matriz[2][j]=matriz[2][j]/indice3;
    }
```

```
     generarmatrizdb("Se vuleve 1 la posición [3][3]",matriz);
     double valor1_2 = matriz[0][2];
     double valor2_2 = matriz[1][2];

     for(int i=0;i< matrizentrada.length;i++){
         for(int j=2;j<matrizentrada[0].length;j++){
             if (i==2){
                 continue;
             }else if(i==0){
                 matriz[i][j]=matriz[i][j]-(matriz[2][j]*valor1_2);
             }else{
                 matriz[i][j]=matriz[i][j]-(matriz[2][j]*valor2_2);
             }
         }
     }
     generarmatrizdb("Se vuelve 0 el resto de filas en la columna 3", matriz);
     reporte+="<h3>El resultado es: </h3>";
     reporte+="<h3>X = "+matriz[0][3]+ "</h3>";
     reporte+="<h3>Y = "+matriz[1][3]+ "</h3>";
     reporte+="<h3>Z = "+matriz[2][3]+ "</h3>";
     escribir("\nEl resultado es:");
     escribir("X = " + matriz[0][3]);
     escribir("Y = "+ matriz[1][3]);
     escribir("Z = "+matriz[2][3]);

     }else{
         escribir("La matriz debe ser de 3*4 para poder resolverla");
     }
}
```

- **LeermatrizGauss:** Lee la matriz ingresada para resolver.

```
private int[][] leermatrizGauss(String ruta) {
    try{
        Scanner leergauss = new Scanner(new FileReader(ruta));
        int[][] matriz = new int[3][4];
        int i = 0;
        while(leergauss.hasNextLine()){
            String linea[] = leergauss.nextLine().split(",");
            for (int j=0;j<4;j++){
                matriz[i][j]= Integer.parseInt(linea[j]);
                escribirsinsalto(matriz[i][j]+"\t");
            }
            escribir("");
            i++;
        }
        generarmatrizint("Matriz Ingresada",matriz);
        return matriz;
    }catch(FileNotFoundException e){
        escribir("No se encuentra el archivo");
        e.printStackTrace();
        return null;
    }
}
```

- **Generarmatrizchar:** Agrega a la variable reporte una matriz char.

```
private void generarmatrizchar(String nombre, char matriz[][]){
    reporte +="<h1>"+nombre+"</h1>\n" +
            "<center><table border=\"2\">\n";
    for(int i=0;i< matriz.length;i++){
        reporte+="<tr>\n";
        for(int j=0;j<matriz[0].length;j++){
            reporte+="<td>"+matriz[i][j]+"</td>";
        }
        reporte+="</tr>";
    }
    reporte+="</table></center>";
}
```

- **Generarmatrizint:** Agrega a la variable reporte una matriz int.

```
private void generarmatrizint(String nombre, int matriz[][]){
    reporte +="<h1>"+nombre+"</h1>\n" +
            "<center><table border=\"2\">\n";
    for(int i=0;i< matriz.length;i++){
        reporte+="<tr>\n";
        for(int j=0;j<matriz[0].length;j++){
            reporte+="<td>"+matriz[i][j]+"</td>";
        }
        reporte+="</tr>";
    }
    reporte+="</table></center>";
}
```

- **Generarmatrizdb:** Agrega a la variable reporte una matriz double.

```
private void generarmatrizdb(String nombre, double matriz[][]){
    reporte +="<h1>"+nombre+"</h1>\n" +
            "<center><table border=\"2\">\n";
    for(int i=0;i< matriz.length;i++){
        reporte+="<tr>\n";
        for(int j=0;j<matriz[0].length;j++){
            reporte+="<td>"+matriz[i][j]+"</td>";
        }
        reporte+="</tr>";
    }
    reporte+="</table></center>";
}
```

- **Generarreporte:** Genera el reporte html de la opción seleccionada.

```
private void generarreporte(String titulo, String ruta){
    String html = "<html>\n" +
            "<head>\n" +
            "<link href=\"estilo.css\" rel=\"Stylesheet\" type=\"text/css\">\n" +
            "<title>"+titulo+"</title>\n" +
            "</head>\n" +
            "<body>\n"+ reporte+"\n</body>\n" +
            "</html>";

    try {
        BufferedWriter escribirreporte = new BufferedWriter(new FileWriter("Reportes/"+ruta+".html",false));
        escribirreporte.write(html);
        escribirreporte.close();
        escribir("Se generó el reporte \""+ruta+".html\" en la carpeta Reportes");
        reporte="";
    }catch(IOException e){
        escribir("Ocurrio un error al escribir el reporte");
    }
}
```

- **Variables Globales:**

```
public String reporte;
public int filasde;
public int columnasde;
```

- **Reporte:** Almacena el reporte en html para después generarlo.
- **Filasde:** Almacena el número de filas de la matriz Descifrar.
- **columnasde:** Almacena el número de columnas de la matriz Descifrar.