UNIVERSIDAD REFAEL BELLOSO CHACIN
FACULTAD DE INGENIERIA
ESCUELA DE INFORMATICA
CATEDRA: PROGRAMACION I
Profa.: Dra. MARIA EUGENIA FOSSI MEDINA

GUIA DE ESTUDIO.

INTRODUCCIÓN: Una parte importante de la algoritmia es la forma como los datos son manipulados, a tal fin, estos pueden ser trabajados de muchas maneras, tal como las variables y las constantes, la cual es la manipulación básica de los datos. Por otra parte, la manipulación se puede realizar por medio de las estructuras de datos, lo cual es una colección de datos organizados de un modo particular.

Las estructuras de datos pueden ser de dos tipos:

- **Estructuras de Datos Estáticas:** Son aquellas en las que se asignan una cantidad fija de memoria cuando se declara la variable.
- **Estructuras de Datos Dinámicas:** Son aquellas estructuras que necesitan que su espacio de memoria crezca o se reduzca a medida que el programa se va ejecutando.

Esta tipología a su vez se ramifica de la siguiente manera:

Estáticas	Array (vectores y matrices) Registros (record) Ficheros o Archivos (file)	
Dinámicas	Lineales	Pilas Colas Listas Enlazadas
	No Lineales	Arboles Grafos

En esta guía, se hará referencia a los arreglos (array), que como se puede observar en la tabla superior, pertenece a las estructuras de datos estáticas.

ARREGLOS (ARRAY): Es un conjunto finito y fijo de elementos del mismo tipo, que se hallan almacenados en posiciones contiguas y consecutivas de memoria. Es una lista lineal (Gonzalo P. Laporta). Por otra parte, Correa Uribe dice que un arreglo es un conjunto de valores o cantidades homogéneas, que por sus cualidades (naturaleza) se comportan de idéntica forma o deben ser tratados en forma similar. A tal fin, se puede decir que un arreglo es una estructura de datos lineal

que puede presentar una o más dimensiones (subíndices) y permite almacenar y manipular varios datos de un mismo tipo. Un arreglo presenta un único nombre para todas sus posiciones de memoria, y estas posiciones se van a identificar por un subíndice particular el cual determina su posición relativa en la memoria.

El nombre que se le asignará al arreglo deberá presentar las mismas características de las variables y en cuanto al subíndice, este deberá ser un número entero positivo que dependiendo del lenguaje de programación, comenzará por cero o por uno. Cabe destacar que la cantidad de índices que tenga el arreglo, indicará el nombre del arreglo. Por ejemplo: cuando el arreglo tiene un solo subíndice, este se llama *unidimensional;* cuando tiene dos subíndices, *bidimensional;* tres subíndices, *tridimensional*; y así sucesivamente.

Otro elemento que se debe tomar en cuanta con respecto a los arreglos, es la *longitud* o el *tamaño* del mismo. En este sentido se puede decir que es el número de elementos que puede manejar o manipular el arreglo. Cuando el arreglo tiene un solo subíndice, simplemente es el último valor del mismo, pero cuando se tiene más de un subíndice, la longitud es el producto de los valores máximos de cada subíndice. Los arreglos cumplen con tres características básicas: a) Almacenar los elementos del arreglo en posiciones continuas de memoria; b) Tener un único nombre de variable que representa a todos los elementos y éstos a su vez se diferencian por un subíndice y c) Acceso directo o aleatorio a los elementos individuales del arreglo.

En cuanto a su utilización, estos pueden ser para varios propósitos ya que permiten agrupar variables relacionadas. Por ejemplo, se puede utilizar un arreglo para almacenar las notas de cada alumno de una sección o para almacenar el precio de cada libro de una librería, entre otras cosas. La ventaja principal de este tipo de estructura, es que permite organizar y clasificar los datos de forma tal que su manipulación sea mucho más fácil.

ARREGLOS UNIDIMENSIONALES (VECTORES): Según Joyanes Aguilar, un arreglo unidimensional es un tipo de dato estructurado compuesto de un número de elementos finito, tamaño fijo y elementos homogéneos. *Finito* indica que hay un último elemento, *tamaño fijo* significa que el tamaño del arreglo debe ser conocido en tiempo de compilación, *homogéneo* significa que todos los elementos son del mismo tipo. Los elementos del arreglo se almacenan en posiciones contiguas de memoria, a cada una de las cuales se puede acceder directamente. Por ejemplo: Suponiendo que se quiere almacenar las temperaturas relacionadas a 50 observaciones de un experimento científico:

Temperaturas	Temperaturas[1]	25.5
	Temperaturas[2]	24.7
	Temperaturas[3]	15.8
	..	. .
	Temperaturas[50]	26.9

Tomando el ejemplo anterior, se puede decir:

Nombre del Vector	Temperaturas
Subíndice	[1], [2], [3], ..., [50]
Contenido	Temperaturas[2] = **24.7**

Se puede observar que en esta estructura, la cual tiene un único nombre (Temperaturas) se están almacenando 50 temperaturas de valores diferentes y un único tipo de dato (real).

Para declarar un arreglo unidimensional en C++, se debe realizar de la siguiente manera:

tipo nombre_arreglo[tamaño];

Donde:

tipo	Declara el tipo básico del arreglo, el cual determina el tipo de cada elemento contenido en el arreglo.
Nombre_arreglo	Es el nombre que se le dará al arreglo, el cual debe tener las mismas características del nombre que se le da a las variables.
[ ]	Indica que lo que se esta declarando es un arreglo unidimensional.
tamaño	Es el numero de elementos que tendrá el arreglo.

Para declarar un arreglo unidimensional en java, se debe realizar de la siguiente manera:

tipo nombre_arreglo[ ] = new tipo[tamaño];

Donde:

tipo	Declara el tipo básico del arreglo, el cual determina el tipo de cada elemento contenido en el arreglo.
Nombre_arreglo	Es el nombre que se le dará al arreglo, el cual debe tener las mismas características del nombre que se le da a las variables.
[ ]	Indica que lo que se esta declarando es un arreglo

	unidimensional.
new	Esta sentencia se utiliza para declarar un objeto. En java, los arreglos se implementa como objeto.
tamaño	Es el numero de elementos que tendrá el arreglo.

Por ejemplo:

int lista[10]; (C++)

O

int lista[ ] = new int[10]; (JAVA)

En este caso se esta declarando un arreglo unidimensional tipo entero, de nombre “lista” y el cual presenta 10 elementos. Es importante destacar que en tanto en C++ como en Java el subíndice comienza en cero y termina en n – 1, es decir, tomando el ejemplo anterior, el ultimo subíndice es 9. Para explicarlo mejor, se graficará el vector:

lista

Dato	23	43	12	34	32	56	12	23	34	76
Subíndice	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

OPERACIONES CON LOS VECTORES:

- *RECORRIDO:* Es el tratamiento secuencial del vector, mediante el cual se accede sucesiva y consecutivamente a los contenidos de cada uno de los elementos del mismo. Este proceso se realiza cada vez que se va a trabajar con el arreglo.

```
// EN C++
#include <iostream>

using namespace std;

main()
{    int i;
     int lista[25];
     cout << "INGRESE LOS DATOS:" << endl;
     for(i=0; i<25; i++)
     {  cout << "INGRESE EL DATO " << i << " = ";
        cin >> lista[i];
     }
 }
```

```
//EN JAVA
public class ejemplo
{
  public static void main(String args[ ])
  {
     int i;
     int lista[ ] = new int[25];
     System.out.println("INGRESE LOS DATOS:");
     for(i=0; i<25; i++)
     {
        System.out.print("INGRESE EL DATO " + i + " = ");
        lista[i] = Integer.parseInt(KbInput.read( ));
     }
  }
}
```

- *INSERCIÓN:* Consiste en introducir en el arreglo el valor de un elemento, bien sea en una celda que halla quedado vacía o para sustituir el valor de una cada ocupada.

```
//EN C++
#include <iostream>

using namespace std;

main()
{   int i,n;
    int lista[25];
    cout << "INGRESE LOS DATOS:" << endl;
    for(i=0; i<25; i++)
    {  cout << "INGRESE EL DATO " << i << " = ";
       cin >> lista[i];
    }
    do{ cout << "ING. LA POSIC. DE LA CELDA A GUARDAR = ";
      cin >> n;
    }while((n<0) || (n>=25));
    cout << "ING. EL VALOR A GUARDAR = ";
    cin >> lista[n];
}

//EN JAVA
public class ejemplo
{
```

```
  public static void main(String args[ ])
  {
     int i,n;
     int lista[ ] = new int[25];
     System.out.println("INGRESE LOS DATOS:");
     for(i=0; i<25; i++)
     {
        System.out.print("INGRESE EL DATO " + i + " = ");
        lista[i] = Integer.parseInt(KbInput.read( ));
     }
     do {
       System.out.print("ING. LA POSIC. DE LA CELDA A GUARDAR: ");
       n = Integer.parseInt(KbInput.read( ));
     } while((n<0) || (n>=25));
     System.out.print("ING. EL VALOR A GUARDAR: ");
     lista[n] = Integer.parseInt(KbInput.read( ));
  }
}
```

- *BORRADO:* Consiste en eliminar del arreglo el valor de un elemento. El borrado se puede realizar de dos formas: a) introduciendo, en la celda de memoria correspondiente, una señal de borrado en lugar del dato almacenado; y b) eliminando totalmente su contenido, mediante el desplazamiento de los elementos del vector para ocupar el espacio borrado, si el elemento en cuestión no es el último del vector, el último elemento del vector queda con una señal de borrado.

SUPONIENDO QUE LA SEÑAL DE BORRADO SEA EL CERO (0) Y QUE SE QUIERE BORRAR EL ELEMENTO NUMERO 6.

```
//EN C++
#include <iostream>

using namespace std;

main()
{  int i,n;
   int lista[25];
   cout << "INGRESE LOS DATOS:" << endl;
   for(i=0; i<25; i++)
   {   cout << "INGRESE EL DATO " << i << " = ";
       cin >> lista[i];
   }
   do{ cout << "ING. LA POSIC. DE LA CELDA A BORRAR = ";
        cin >> n;
```

```
    }while((n<0) || (n>=25));
    lista[n] = 0;
}

//EN JAVA
public class ejemplo
{
  public static void main(String args[ ])
  {
    int i,n;
    int lista[ ] = new int[25];
    System.out.println("INGRESE LOS DATOS:");
    for(i=0; i<25; i++)
    {
      System.out.print("INGRESE EL DATO " + i + " = ");
      lista[i] = Integer.parseInt(KbInput.read( ));
    }
    do {
      System.out.print("ING. LA POSIC. DE LA CELDA A BORRAR: ");
      n = Integer.parseInt(KbInput.read( ));
    } while((n<0) || (n>=25));
    lista[n] = 0;
  }
}
```

SUPONIENDO QUE SE VA A ELIMINAR EL MISMO ELEMENTO DESPLAZANDO LOS QUE ESTAN POR ENCIMA DE EL Y LA SEÑAL DE BORRADO SEA CERO.

```
//EN C++
#include <iostream>

using namespace std;

main()
{  int i,n,j;
   int lista[25];
   cout << "INGRESE LOS DATOS:" << endl;
   for(i=0; i<25; i++)
   {   cout << "INGRESE EL DATO " << i << " = ";
       cin >> lista[i];
   }
   do{ cout << "ING. LA POSIC. DE LA CELDA A BORRAR = ";
       cin >> n;
   }while((n<0) || (n>=24));
   for(i=n; i<25; i++)
```

```
   {   j = i + 1;
       lista[i] = lista[j];
   }
   lista[24] = 0;
}

//EN JAVA
public class ejemplo
{
  public static void main(String args[ ])
  {
     int i,n,j;
     int lista[ ] = new int[25];
     System.out.println("INGRESE LOS DATOS:");
     for(i=0; i<25; i++)
     {
        System.out.print("INGRESE EL DATO " + i + " = ");
        lista[i] = Integer.parseInt(KbInput.read( ));
     }
     do {
       System.out.print("ING. LA POSIC. DE LA CELDA A BORRAR: ");
       n = Integer.parseInt(KbInput.read( ));
     } while((n<0) || (n>=24));
     for(i=n; i<25; i++)
     {
        j = i + 1;
        lista[i] = lista[j];
     }
     lista[24] = 0;
  }
}
```

- *BUSQUEDA:* Consiste en realizar un recorrido del vector, comenzando de su posición más baja de memoria (en este caso desde cero), a fin de localizar en el arreglo un dato determinado. En este recorrido, se puede dar el caso de que no exista el dato o en caso contraria que este más de uno.

```
//EN C++
#include <iostream>

using namespace std;

main()
{  int i,valor,c=0;
```

```
  int lista[25];
  cout << "INGRESE LOS DATOS:" << endl;
  for(i=0; i<25; i++)
  {
    cout << "INGRESE EL DATO " << i << " = ";
    cin >> lista[i];
  }
  cout << "ING. EL VALOR A BUSCAR = ";
  cin >> valor;
  for(i=0; i<25; i++)
  {
    if(lista[i] == valor)
    {
      c++;
      cout << "SE ENCUENTRA EN LA CELDA " << i << endl;
    }
  }
  if(c == 0)
    cout << "NO EXISTE NINGUN ELEM. CON ESE VALOR" << endl;
  else
    cout << "EXISTEN " << c << " CON ESE VALOR" << endl;
  system("PAUSE");
}

//EN JAVA
public class ejemplo
{
  public static void main(String args[ ])
  {
    int i,valor,c=0;
    int lista[ ] = new int[25];
    System.out.println("INGRESE LOS DATOS:");
    for(i=0; i<25; i++)
    {
      System.out.print("INGRESE EL DATO " + i + " = ");
      lista[i] = Integer.parseInt(KbInput.read( ));
    }
    System.out.print("ING. EL VALOR A BUSCAR: ");
    valor = Integer.parseInt(KbInput.read( ));
    for(i=0; i<25; i++)
    {
      if(lista[i] == valor)
      {
        c++;
```

```
            System.out.println(valor + “SE ENCUENTRA EN LA CELDA “+i);
          }
        }
        if(c == 0)
          System.out.println(“NO EXISTE NINGUN ELEM. CON ESE VALOR”);
        else
          System.out.println(“EXISTEN “ + c + ” CON ESE VALOR”);
    }
}
```

- *ORDENACION:* Consiste en reorganizar el contenido de cada uno de los elementos del vector según una secuencia determinada.

ORDENACION ASCENDENTE

```
//EN C++
#include <iostream>

using namespace std;

main()
{   int i,j,aux;
    int lista[25];
    cout << "INGRESE LOS DATOS:" << endl;
    for(i=0; i<25; i++)
    {
        cout << "INGRESE EL DATO " << i << " = ";
        cin >> lista[i];
    }
    aux = 0;
    for(i=0; i<(25 - 1); i++)
    {  for(j=i; j<25; j++)
       {  if(lista[i] > lista[j])
          {  aux = lista[i];
             lista[i] = lista[j];
             lista[j] = aux;
          }
       }
    }
    cout << "LA ORDENACION ASCENDENTE ES: " << endl;
    for(i=0; i<25; i++)
       cout << "ELEMENTO "<< i << " ES = " << lista[i] << endl;
    system("PAUSE");
}
```

```
//EN JAVA
public class ejemplo
{
  public static void main(String args[ ])
  {
     int i,j,aux;
     int lista[ ] = new int[25];
     System.out.println(“INGRESE LOS DATOS:”);
     for(i=0; i<25; i++)
     {
        System.out.print(“INGRESE EL DATO “ + i + ” = “);
        lista[i] = Integer.parseInt(KbInput.read( ));
     }
     aux = 0;
     for(i=0; i<(25 – 1); i++)
     {
        for(j=i; j<25 ; j++)
        {
           if(lista[i] > lista[j])
           {
              aux = lista[i];
              lista[i] = lista[j];
              lista[j] = aux;
           }
        }
     }
     System.out.println(“LA ORDENACION ASCENDENTE ES: ”);
     for(i=0; i<25; i++)
        System.out.println(“ELEMENTO “+i+ “ ES = “+lista[i]);
  }
}
```

ORDENACION DESCENDENTE

```
//EN C++
#include <iostream>

using namespace std;

main()
{  int i,j,aux;
   int lista[25];
   cout << "INGRESE LOS DATOS:" << endl;
   for(i=0; i<25; i++)
   {
```

```
    cout << "INGRESE EL DATO " << i << " = ";
    cin >> lista[i];
  }
  aux = 0;
  for(i=0; i<(25 - 1); i++)
  {
    for(j=i; j<25 ; j++)
    {
       if(lista[i] < lista[j])
       {
          aux = lista[i];
          lista[i] = lista[j];
          lista[j] = aux;
       }
    }
  }
  cout << "LA ORDENACION DESCENDENTE ES: " << endl;
  for(i=0; i<25; i++)
    cout << "ELEMENTO " << i << " ES = " << lista[i] << endl;
  system("PAUSE");
}

//EN JAVA
public class ejemplo
{
  public static void main(String args[ ])
  {
    int i,j,aux;
    int lista[ ] = new int[25];
    System.out.println("INGRESE LOS DATOS:");
    for(i=0; i<25; i++)
    {
      System.out.print("INGRESE EL DATO " + i + " = ");
      lista[i] = Integer.parseInt(KbInput.read( ));
    }
    aux = 0;
    for(i=0; i<(25 – 1); i++)
    {
      for(j=i; j<25 ; j++)
      {
         if(lista[i] < lista[j])
         {
            aux = lista[i];
            lista[i] = lista[j];
```

```
                lista[j] = aux;
            }
        }
    }
    System.out.println("LA ORDENACION DESCENDENTE ES: ");
    for(i=0; i<25; i++)
       System.out.println("ELEMENTO "+i+ " ES = "+lista[i]);
  }
}
```

Para ver mejor como se utilizan los arreglos en java, se va a realizar el siguiente ejemplo: Calcular el promedio de 25 notas e indicar cuantas están por debajo del mismo.

```
//EN C++
#include <iostream>

using namespace std;

main()
{
  int i,ndp = 0;
  float prom, sum = 0;
  float nota[25];
  cout << "INGRESE LAS NOTAS DE LOS ALUMNOS:" << endl;
  for(i=0; i<25; i++)
  {
    do{
        cout << "INGRESE LA NOTA "<< i <<" = ";
        cin >> nota[i];
    }while((nota[i]<0) || (nota[i]>20));
    sum+=nota[i];
  }
  prom = sum / 25;
  for(i=0; i<25; i++)
  {
     if( nota[i] < prom)
       ndp++;
  }
  cout << "EL PROMEDIO DE LAS 25 NOTAS = " << prom << endl;
  cout << "EXISTEN " << ndp << " MENORES AL PROMEDIO" << endl;
  system("PAUSE");
}
```

```
//EN JAVA
public class notas
{
  public static void main(String args[ ])
  {
     int i,ndp = 0;
     float prom, sum = 0;
     float nota[ ] = new float[25];
     System.out.println("INGRESE LAS NOTAS DE LOS ALUMNOS:");
     for(i=0; i<25; i++)
     {
        do {
          System.out.print("INGRESE LA NOTA "+ i +" = ");
          nota[i] = Float.parseFloat(KbInput.read( ));
        }while((nota[i]<0) || (nota[i]>20));
        sum+=nota[i];
     }
     prom = sum / 25;
     for(i=0; i<25; i++)
     {
        if( nota[i] < prom)
           ndp++;
     }
     System.out.println("EL PROMEDIO DE LAS 25 NOTAS = " + prom);
     System.out.println("EXISTEN "+ ndp + " MENORES AL PROMEDIO");
  }
}
```

ARREGLOS BIDIMENSIONALES (MATRICES): Según Joyanes Aguilar es un vector de vectores. Es por tanto un conjunto de elementos del mismo tipo en el que el orden de los componentes es significativo y en el que se necesitan especificar dos subíndices para poder identificar a cada elemento del arreglo. Por otra parte, Pascual Laporta lo define como: un conjunto de datos, del mismo tipo, estructurado de tal forma que se precisan dos índices para referenciar cada uno de sus elementos. Por último, Schildt dice que un arreglo bidimensional es en esencia una lista de arreglos unidimensionales.

Tomando las definiciones anteriores, se puede decir que un arreglo bidimensional es una estructura que permite el manejo de varios datos de un mismo tipo a la vez. El mismo se representa por medio de filas y columnas, por tal motivo se necesita el manejo de dos subíndices (uno para la fila, el cual se coloca primero, y el otro para la columna), lo cual hace la idea de que se este trabajando con una matriz.

Ejemplo: Suponga que se tiene un arreglo **B** de cuatro filas y tres columnas. En este caso, el termino general seria **B(I,J)**; donde **B** es el nombre del arreglo, **I** y **J** son los subíndices que indican la fila y la columna respectivamente del elemento.

Representación:

FILAS	**COLUMNAS**		
	1	**2**	**3**
1	B(1,1)	B(1,2)	B(1,3)
2	B(2,1)	B(2,2)	B(2,3)
3	B(3,1)	B(3,2)	B(3,3)
4	B(4,1)	B(4,2)	B(4,3)

En la tabla, el índice **I(1,2,3,4)** representa cada una de las filas del arreglo, mientras que el índice **J(1,2,3)** representa cada una de las columnas. Es importante resaltar que tanto I como J son variables y su identificador puede cambiar de nombre. Por otra parte, para saber cual es el número total de elementos del arreglo, se deberá multiplicar el número de filas por el número de columnas. En este sentido, se puede decir que el arreglo B trabaja con un total de 12 elementos.

B(4,3) = 4 * 3 = 12

Al igual que en los arreglos unidimensionales, cuando se va ha trabajar con Java los subíndices comienzan en cero y terminan en el subíndice –1. Para verlo mejor, observe la representación:

FILAS	**COLUMNAS**		
	0	**1**	**2**
0	B(0,0)	B(0,1)	B(0,2)
1	B(1,0)	B(1,1)	B(1,2)
2	B(2,0)	B(2,1)	B(2,2)
3	B(3,0)	B(3,1)	B(3,2)

Ahora bien, para declarar un arreglo bidimensional en C++ o en Java, se debe realizar de la siguiente manera:

tipo nombre_arreglo [tamaño_f] [tamaño_c]; //EN C++

O

tipo nombre_arreglo[ ][ ] = new tipo[tamaño_f] [tamaño_c]; //EN JAVA

Donde:

Tipo	Declara el tipo básico del arreglo, el cual determina el tipo de cada elemento contenido en el arreglo.
Nombre_arreglo	Es el nombre que se le dará al arreglo, el cual debe tener las mismas características del nombre que se le da a las variables.
[ ][ ]	Indica que lo que se esta declarando es un arreglo bidimensional.
New (solo en Java)	Esta sentencia se utiliza para declarar un objeto. En java, los arreglos se implementa como objeto.
Tamaño_f	Es el número de filas que tendrá el arreglo.
Tamaño_c	Es el número de columnas que tendrá el arreglo.

Ejemplo: suponga que se quiere declarar un arreglo bidimensional llamado tabla de 4 filas y 3 columnas:

```
int tabla [4][3];  //EN C++
```

O

```
int tabla[ ][ ] = new int[4][3]; //EN JAVA
```

OPERACIONES CON LAS MATRICES:

- *RECORRIDO:* Al igual que en los arreglos unidimensionales, en las matrices, se utiliza el recorrido para acceder a los elementos del arreglo. Sin embargo, es importante resaltar que el recorrido se realiza por filas, es decir, primero se recorre la fila cero (0), luego la fila (1) y así sucesivamente.

```
//EN C++
#include <iostream>

using namespace std;

main()
{   int i,j;
    int tabla[4][3];
    cout << "INGRESE LOS DATOS:" << endl;
    for(i=0; i<4; i++)
    {   for(j=0; j<3; j++)
        {
            cout << "INGRESE EL DATO " << i << "," << j << " = ";
            cin >> tabla[i][j];
        }
```

```
   }
}

//EN JAVA
public class ejemplo
{
  public static void main(String args[ ])
  {
     int I,j;
     int tabla[ ][ ] = new int[4][3];
     System.out.println("INGRESE LOS DATOS:");
     for(i=0; i<4; i++)
     {
        for(j=0; j<3; j++)
       {
          System.out.print("INGRESE EL DATO " + i + "," + j + " = ");
          tabla[i][j] = Integer.parseInt(KbInput.read( ));
       }
     }
  }
}
```

Representación del arreglo **tabla[4][3]**: La primera fila a ingresar es la F1, luego la F2 y así sucesivamente:

FILAS		**COLUMNAS**		
		0	**1**	**2**
F1	**0**	3 →	5 →	1
F2	**1**	5 →	9 →	0
F3	**2**	8 →	4 →	4
F4	**3**	10 →	6 →	5

- *INSERCIÓN:*

```
//EN C++
#include <iostream>

using namespace std;

main()
{   int i,j,ni,nj;
    int tabla[4][3];
    cout << "INGRESE LOS DATOS:" << endl;
    for(i=0; i<4; i++)
```

```
    {   for(j=0; j<3; j++)
        {
            cout << "INGRESE EL DATO " << i << "," << j << " = ";
            cin >> tabla[i][j];
        }
    }
    do{  cout << "ING. LA FILA DE LA CELDA A GUARDAR = ";
         cin >> ni;
    }while((ni<0) || (ni>=4));
    do{  cout << "ING. LA COLUM. DE LA CELDA A GUARDAR = ";
         cin >> nj;
    }while((nj<0) || (nj>=3));
    cout << "ING. EL VALOR A GUARDAR = ";
    cin >> tabla[ni][nj];
}

//EN JAVA
public class ejemplo
{
  public static void main(String args[ ])
  {
     int i,j,ni,nj;
     int tabla[ ][ ] = new int[4][3];
     System.out.println("INGRESE LOS DATOS:");
     for(i=0; i<4; i++)
     {
         for(j=0; j<3; j++)
        {
            System.out.print("INGRESE EL DATO " + i + "," + j + " = ");
            tabla[i][j] = Integer.parseInt(KbInput.read( ));
        }
     }
     do {
       System.out.print("ING. LA FILA DE LA CELDA A GUARDAR: ");
       ni = Integer.parseInt(KbInput.read( ));
     } while((ni<0) || (ni>=4));
     do {
       System.out.print("ING. LA COLUM. DE LA CELDA A GUARDAR: ");
       nj = Integer.parseInt(KbInput.read( ));
     } while((nj<0) || (nj>=3));
     System.out.print("ING. EL VALOR A GUARDAR: ");
     tabla[ni] [nj] = Integer.parseInt(KbInput.read( ));
  }
}
```

- *BORRADO:*

```
//EN C++
#include <iostream>

using namespace std;

main()
{  int i,j,ni,nj;
   int tabla[4][3];
   cout << "INGRESE LOS DATOS:" << endl;
   for(i=0; i<4; i++)
   {  for(j=0; j<3; j++)
      {
         cout << "INGRESE EL DATO " << i << "," << j << " = ";
         cin >> tabla[i][j];
      }
   }
   do{ cout << "ING. LA FILA DE LA CELDA A BORRAR = ";
         cin >> ni;
   }while((ni<0) || (ni>=4));
   do{ cout << "ING. LA COLUM. DE LA CELDA A BORRAR = ";
         cin >> nj;
   }while((nj<0) || (nj>=3));
   tabla[ni][nj] = 0;
}

//EN JAVA
public class ejemplo
{
  public static void main(String args[ ])
  {
     int i,j,ni,nj;
     int tabla[ ][ ] = new int[4][3];
     System.out.println("INGRESE LOS DATOS:");
     for(i=0; i<4; i++)
     {
        for(j=0; j<3; j++)
       {
          System.out.print("INGRESE EL DATO " + i + "," + j + " = ");
```

```
            tabla[i][j] = Integer.parseInt(KbInput.read( ));
          }
      }
      do {
        System.out.print("ING. LA FILA DE LA CELDA A BORRAR: ");
        ni = Integer.parseInt(KbInput.read( ));
      } while((ni<0) || (ni>=4));
      do {
        System.out.print("ING. LA COLUM. DE LA CELDA A BORRAR: ");
        nj = Integer.parseInt(KbInput.read( ));
      } while((nj<0) || (nj>=3));
      tabla[ni] [nj] = 0;
   }
}
```

- *BUSQUEDA:*

```
//EN C++
#include <iostream>

using namespace std;

main()
{  int i,j,valor,ci=0;
   int tabla[4][3];
   cout << "INGRESE LOS DATOS:" << endl;
   for(i=0; i<4; i++)
   {  for(j=0; j<3; j++)
      {  cout << "INGRESE EL DATO " << i << "," << j << " = ";
         cin >> tabla[i][j];
      }
   }
   cout << "ING. EL VALOR A BUSCAR = ";
   cin >> valor;
   for(i=0; i<4; i++)
   {  for(j=0; j<3; j++)
      {   if(tabla[i][j] == valor)
          {  ci++;
             cout << valor << " ESTA EN LA CELDA "<< i << "," << j << endl;
          }
      }
   }
   if(ci == 0)
     cout << "NO EXISTE NINGUN ELEM. CON ESE VALOR" << endl;
```

```
   else
    cout << "EXISTEN " << ci << " ELEM. CON ESE VALOR" << endl;
   system("PAUSE");
}

//EN JAVA
public class ejemplo
{
  public static void main(String args[ ])
  {
     int I,j,valor,ci=0;
     int tabla[ ][ ] = new int[4][3];
     System.out.println("INGRESE LOS DATOS:");

     for(i=0; i<4; i++)
     {
        for(j=0; j<3; j++)
       {
          System.out.print("INGRESE EL DATO " + i + "," + j + " = ");
          tabla[i][j] = Integer.parseInt(KbInput.read( ));
       }
     }
     System.out.print("ING. EL VALOR A BUSCAR: ");
     valor = Integer.parseInt(KbInput.read( ));
     for(i=0; i<4; i++)
     {
        for(j=0; j<3; j++)
       {
         if(tabla[i][j] == valor)
        {
           c++;
           System.out.println(valor + "ESTA EN LA CELDA "+ i + "," + j);
        }
       }
     }
     if(c == 0)
       System.out.println("NO EXISTE NINGUN ELEM. CON ESE VALOR");
     else
       System.out.println("EXISTEN " + c + " ELEM. CON ESE VALOR");
  }
}
```

- *ORDENACION:* Para realizar una ordenación total de la matriz, se deberá operar del modo siguiente:

- Almacenar los valores de la matriz en un vector auxiliar, cuyo número de elementos sea igual al de la matriz.
- Ordenar el vector por cualquier método.
- Almacenar los valores ordenados del vector en la matriz mediante un recorrido por la misma.

El proceso de cambio de matriz a vector y viceversa es:

```
//EN C++
#include <iostream>

using namespace std;

main()
{   int i,j,k;
    int tabla[4][3];
    int aux[12];
    cout << "INGRESE LOS DATOS:" << endl;
    for(i=0; i<4; i++)
    {  for(j=0; j<3; j++)
       {  cout << "INGRESE EL DATO " << i << "," << j << " = ";
          cin >> tabla[i][j];
       }
    }
    // LLEVAR DE LA MATRIZ AL VECTOR
    k=0;
    for(i=0; i<4; i++)
    {  for(j=0; j<3; j++)
       {  aux[k] = tabla[i][j];
          k++;
       }
    }
    // LUEGO DE ESTE PASO SE REALIZA EL ORDENAMIENTO
    // LLEVAR DEL VECTOR A LA MATRIZ
    k=0;
    for(i=0; i<4; i++)
    {  for(j=0; j<3; j++)
       {  tabla[i][j] = aux[k];
          k++;
       }
    }
}

//EN JAVA
```

```
public class ejemplo
{
  public static void main(String args[ ])
  {
     int I,j,k;
     int tabla[ ][ ] = new int[4][3];
     int aux[ ] = new int[12];
     System.out.println("INGRESE LOS DATOS:");
     for(i=0; i<4; i++)
     {
        for(j=0; j<3; j++)
       {
          System.out.print("INGRESE EL DATO " + i + "," + j + " = ");
          tabla[i][j] = Integer.parseInt(KbInput.read( ));
       }
     }

// LLEVAR DE LA MATRIZ AL VECTOR
     k=0;
     for(i=0; i<4; i++)
     {
        for(j=0; j<3; j++)
       {
          aux[k] = tabla[i][j];
          k++;
       }
     }

// LUEGO DE ESTE PASO SE REALIZA EL ORDENAMIENTO
// LLEVAR DEL VECTOR A LA MATRIZ
     k=0;
     for(i=0; i<4; i++)
     {
        for(j=0; j<3; j++)
       {
          tabla[i][j] = aux[k];
          k++;
       }
     }
  }
}
```

Ejemplo: Realizar un programa que lea dos matrices de m*n y posteriormente realice la suma de ellas.

```
//EN C++
#include <iostream>

using namespace std;

main()
{  int i,j,f,c;
   do{ cout << "INGRESE EL NUMERO DE FILAS = ";
       cin >> f;
   }while(f<0);
   do{ cout << "INGRESE EL NUMERO DE COLUMNAS = ";
       cin >> c;
   }while(c<0);
   int matriz1[f][c];
   int matriz2[f][c];
   int matrizS[f][c];
   cout << "INGRESE LOS DATOS DE LA MATRIZ 1:" << endl;
   for(i=0; i<f; i++)
   {   for(j=0; j<c; j++)
       {   cout << "INGRESE EL DATO " << i << "," << j << " = ";
           cin >> matriz1[i][j];
       }
   }
   cout << "INGRESE LOS DATOS DE LA MATRIZ 2:" << endl;
   for(i=0; i<f; i++)
   {   for(j=0; j<c; j++)
       {   cout << "INGRESE EL DATO " << i << "," << j << " = ";
           cin >> matriz2[i][j];
       }
   }
   cout << "MATRIZ SUMA ES:" << endl;
   for(i=0; i<f; i++)
   {  for(j=0; j<c; j++)
      {  matrizS[i][j] = matriz1[i][j] + matriz2[i][j];
         cout << "ELEMENTO " << i << "," << j << " = " << matrizS[i][j] << endl;
      }
   }
   system("PAUSE");
}


//EN JAVA
public class SumaMatriz
{
  public static void main(String args[ ])
```

```
 {
   int l,j,f,j;
   do {
     System.out.print("INGRESE EL NUMERO DE FILAS = ");
     f = Integer.parseInt(KbInput.read( ));
   } while(f<0);
   do {
     System.out.print("INGRESE EL NUMERO DE COLUMNAS = ");
     c = Integer.parseInt(KbInput.read( ));
   } while(c<0);

   int matriz1[ ][ ] = new int[f][c];
   int matriz2[ ][ ] = new int[f][c];
   int matrizS[ ][ ] = new int[f][c];

   System.out.println("INGRESE LOS DATOS DE LA MATRIZ 1:");
   for(i=0; i<f; i++)
   {
      for(j=0; j<c; j++)
     {
        System.out.print("INGRESE EL DATO " + i + "," + j + " = ");
        Matriz1[i][j] = Integer.parseInt(KbInput.read( ));
     }
   }
   System.out.println("INGRESE LOS DATOS DE LA MATRIZ 2:");
   for(i=0; i<f; i++)
   {
      for(j=0; j<c; j++)
     {
        System.out.print("INGRESE EL DATO " + i + "," + j + " = ");
        Matriz2[i][j] = Integer.parseInt(KbInput.read( ));
     }
   }
   System.out.println("MATRIZ SUMA ES:");
   for(i=0; i<f; i++)
   {
      for(j=0; j<c; j++)
     {
        matrizS[l][j] = matriz1[l][j] + matriz2[l][j];
        System.out.print("ELEMENTO " + i + "," + j + " = " +  matrizS[l][j]);
     }
   }
  }
}
```

BIBLIOGRAFIA

1.- **Gonzalo Pascual L.**
"Estructura de la Información".
McGrawHill. 1992

2.- **Guillermo Correa U.**
"Desarrollo de Algoritmos y sus Aplicaciones en Basic, Pascal, Cobol y C".
McGrawHill. 1992

3.- **Herbert Schildt**
"C++ Para Programadores".
McGrawHill. 1996

4.- **Herbert Schildt**
"Fundamentos de Programación en JAVA 2".
McGrawHill. 2002

5.- **Herbert Schildt**
"Manual de Referencia JAVA 2".
McGrawHill. 2001

6.- **Luis Joyanes A., Luis Rodríguez B. y Matilde Fernández A.**
"Fundamentos de Programación".
McGrawHill. 1996

7.- **Luis Joyanes A. y Matilde Fernández A.**
"JAVA 2 Manual de Programación".
McGrawHill. 2001