

CAPÍTULO 4. ARREGLOS

La manipulación de datos es una de las principales actividades que realizan los programas, por tal motivo se han creado varias estructuras que permiten manejar los datos de diferentes maneras, pero siempre buscando la optimización en su manejo.

Una de las primeras estructuras de datos que se estudia son los arreglos, por tal motivo dedicamos este capítulo a su estudio, en el mismo cubriremos los conocimientos básicos sobre arreglos.

Ante todo es importante conocer la importancia de los arreglos. Para ello analice el *Algoritmo 5 ALUMNOS FOR* del capítulo 3 de esta guía didáctica. Recuerde que la característica que define al algoritmo anterior es la utilización de la sentencia FOR y cuyo objetivo es tomar cinco veces por teclado (leer) los nombres, apellidos y presentarlos en pantalla. De esta manera se evita la definición de cinco variables para nombres (**n1, n2, n3, ...**) y cinco variables para los apellidos (**a1, a2, a3, ...**) lo que implica un ahorro de espacio en memoria. Pero, ¿qué implicó que solo se defina una variable **nom** para los nombres y una variable **ape** para los apellidos?. Pues bien, el problema principal es que los valores anteriores de las variables en cada repetición del FOR serán sobrescritos por los nuevos valores ingresados, en otras palabras, si se desea ingresar una lista de 1000 nombres, con el algoritmo anterior y luego proceder a presentar los 100 primeros nombres resultaría imposible realizarlo mediante el procedimiento anterior, ya que no se ha guardado más que el último valor que se ingreso.

El siguiente algoritmo está modificado para el propósito antes señalado y nuevamente se tiene que definir las 1000 variables para nombres y 1000 más para apellidos con el fin de no perder valores de las variables, lo que volvería a consumir recursos de memoria y en este acaso a tener un programa con muchas líneas de código.

Algoritmo ALUMNOS FOR CON ARREGLOS

Clase Alumnos

1. Método principal

a. Declaraciones

Variables

n1, n2, n3, ...n1000:Cadena

a1, a2, a3, ...a1000:Cadena

i : Entero

b. Solicitar Nombre y apellido de 1000 alumnos

1. Leer n1, a1

2. Leer n2, a2

3. Leer n3, a3

....



```

1000. Leer n1000, a1000
c. Presentar nombre y apellido de 100 alumnos
    1. Presentar n1, a1
    2. Presentar n2, a2
    3. Presentar n3, a3
    ....
    100. Presentar n100, a100

d. Fin Método principal
Fin Clase AlumnosFor
Fin

```

Para resolver problemas que trabajan con un conjunto de variables del mismo tipo se utiliza arreglos que en si son estructuras de datos que pueden contener información de un mismo tipo. Los arreglos permiten manejar grandes conjuntos de datos solamente con una sola variable y mediante la utilización de índices. En nuestro ejemplo bastaría con declarar *nom[1000]* y *ape[1000]*.

```

Algoritmo ALUMNOS FOR CON ARREGLOS
Clase AlumnosFor
    1. Método principal
        a. Declaraciones
            Variables
                nom[1000], ape[1000]: Cadena
                i : Entero

            b. FOR i = 1; i <= 1000; i ++
                1. Solicitar Nombre y apellido del alumno i
                2. Leer nom[ i ], ape[ i ]
            c. ENDFOR
            d. FOR i = 1; i <= 100; i ++
                1. Presentar nom[ i ], ape[ i ]
            e. ENDFOR
        f. Fin Método principal
Fin Clase AlumnosFor
Fin

```

Existen otros problemas de programación en donde es necesario contar con muchas variables del mismo tipo. Se puede mencionar algunas áreas como: estadísticas, ingeniería civil, ingeniería química, entre otras.

Una vez planteada la necesidad de contar con estructuras de datos, como los arreglos, es necesario pasar a estudiar cómo definir un arreglo de cualquier dimensión.



4.1 Definición



Realice una lectura comprensiva de los acápites del 8.1 “Arreglos unidimensionales” al 8.4 “Arreglos tetradimensionales”, página 164, Capítulo 8 “Arreglos” del texto básico. En el mismo que inicia con las conceptualizaciones de los arreglos unidimensionales, bidimensionales, tridimensionales, etc. Además analice y entienda de cada uno de los ejemplos propuestos del texto básico.

¿Qué le pareció la lectura? ¿Comprendió? Si es así avancemos y repasemos ciertos elementos de la lectura anterior.

Un arreglo es una estructura de datos estática que almacena información de un mismo tipo de dato. Existen diferentes tipos de arreglos:

- Arreglos unidimensionales.
- Arreglos bidimensionales.
- Arreglos tridimensionales.
- Arreglos tetradimensionales.

Es importante que lea detenidamente la sección 8.1 “Arreglos unidimensionales” del texto básico y entienda cual es su estructura. A continuación algunas ideas interesantes sobre el tema:

1. La característica de estos es que el tamaño (**N**) esta representado con un solo valor, por ejemplo:

Producción: Arreglo [30] Entero, en donde N=30

2. **IMPORTANTE:** Además los datos son accedidos por un solo índice que van desde un valor inicial (1) hasta un valor final (**N**). En algunos casos como lo indica el texto básico existen lenguajes de programación (Java, C++, C#, Object-c) que inician desde 0 (cero), en cuyo caso el valor final del arreglo será **N-1**.
3. Los elementos se relacionan con un nombre de la variable y un índice que generalmente va entre corchetes “[]”. Dentro de este pueden ir números o variables enteras. También se pueden realizar operaciones matemáticas como por ejemplo:

`produccion[(i*4)-j+2]`

4. Los arreglos se recorren mediante estructuras repetitivas como el FOR. Revise el apartado 8.1. en *Manejo de elementos del arreglo unidimensional*. En el Algoritmo PRODUCCIÓN 30 Días, observe como se ingresan datos en los arreglos utilizando las sentencias FOR y



dentro la instrucción **Leer produccion[i]**. Así mismo muestra como

se presentan los datos. Esta es la manera general de como se recorre un arreglo para la introducir datos y luego para presentarlos.

Acuda nuevamente al texto básico y lea el acápite 8.2 “Arreglos bidimensionales” para conocer más de los arreglos bidimensionales o matrices. Luego de haber leído podrá darse cuenta que los arreglos bidimensionales se diferencian de los unidimensionales por la disposición de los elementos dentro del arreglo de datos. A continuación algunas ideas a considerar:

1. En la sección 8.2.1 “Ejercicios resueltos para bidimensionales” se muestra que los arreglos bidimensionales están conformados por filas y columnas.
2. Ahora los elementos tienen dos subíndices:

Matriz[filas][columnas]

3. Además el tamaño del arreglo está dado por dos valores **M** que representa el número de filas y **N** el número de columnas. Entonces se dicen que una matriz tiene un orden en dos dimensiones **MXN**. Como por ejemplo

Matriz: Arreglo[4][5] Entero; M=4 y N=5, es decir 20 elementos.

Y por último revise los apartados 8.3 “Arreglos tridimensionales” y 8.4 “Arreglos tetradimensionales” del texto básico para aprender de los arreglos tridimensionales y tetradimensionales. Luego usted podrá concluir que estos tienen las mismas consideraciones que los arreglos unidimensionales y los bidimensionales. Tal como lo explica en el texto básico los tridimensionales son una combinación de los unidimensionales y los bidimensionales. Y los tetradimensionales, en cambio una combinación entre dos bidimensionales.

Una vez analizadas las definiciones, vamos a estudiar las acciones básicas que se pueden realizar con los arreglos en general (independiente de sus dimensiones). Iniciamos con la declaración de arreglos.

4.2 Declaración de arreglos



Realice una lectura comprensiva de las páginas 165, 174, 184 y 189 del texto básico, ahí encontrará como se puede definir los arreglos desde los unidimensionales hasta los tetradimensionales



En el acápite 8.1 “Arreglos unidimensionales” del texto básico se presentan diferentes formas de declaración de arreglos unidimensionales, se podría decir:

la forma básica, y mediante tipos. En resumen la primera forma implica:

Nombre de la variable: Arreglo[Tamaño] Tipo de dato

La segunda,

Declarar Tipo

Variable de tipo: Arreglo[Tamaño] tipo de dato

Declarar variable

Nombre de la variable: Variable de tipo

Las dos formas son válidas, aunque la primera es la mas utilizada por la simplicidad que tiene.

A continuación se presenta algunos fragmentos como ejemplos donde se declaran arreglos en el lenguaje Java. Observe y entienda la parte de la declaración y su inicialización con la palabra reservada **new**.

```
12 public static void main(String[] args) {
13     Scanner lector= new Scanner(System.in);
14     int M,N;
15     int mayor, menor;
16     int[] vector;
17     String[] nombres;
18     int[][] Matriz;
19
20     System.out.println("Introduce el orden de la matriz MXN");
21     System.out.print("Introduce el número de filas M:");
22     M=lector.nextInt(); //Ingreso por teclado del día
23     System.out.print("Introduce el número de columnas N:");
24     N=lector.nextInt(); //Ingreso por teclado del mes
25
26     nombres= new String[10];
27     vector= new int [N];
28     Matriz = new int [M][N];
```

Ilustración 1 Declaración de arreglos (1)

En la Ilustración 56, en la línea 16 podemos ver como se declara un arreglo unidimensional entero y en la línea 27 como se inicializa con los valores **M** y **N** que son ingresados por el usuario. También podemos observar que se declara un arreglo unidimensional de tipo texto y luego en la línea 26 su respectiva creación con una longitud de 10. En la línea 16 se declara un arreglo bidimensional, en la línea 28 se crea con **M** filas y **N** columnas.

De esta forma podemos declarar y crear arreglos de cualquier tipo de dato (entero, real, cadena, carácter). Recuerde que para la creación del arreglo se



ubica el tamaño dentro de los corchetes. Ahora usted tome los ejercicios de la sección 8.1.1 “Ejercicios resueltos para unidimensionales” del texto básico y declare los arreglos de los ejemplos.

En el siguiente programa Java podrá observar como se realiza la creación e inicialización de un arreglo de enteros.

```
52 public void ordenInsercion(){
53
54     int[] E={1,2,3,4,5,6,7,8,9,10};
55     int n=E.length;
56     int index;
57     for (index=0;index<n; index++){
58         int actual=E[index];
59         int x=actual;
60         int posX = desplaVac(E, index, x);
61         E[posX]=actual;
62         printarreglo(E);
63     }
64
65     int a=0;
66 }
67 }
```

Ilustración 2 Declaración de arreglos (2)

En el programa de la Ilustración 57 se encuentra un método de una clase (en el capítulo 6 se hablará mas del tema). Aquí se declara al arreglo inicializando valores en la línea 54. Observe que los **valores de inicialización** van entre llaves y separados por comas. Esta es la manera más común para introducir valores de inicio en los arreglos. Observe cual es la función de línea 55. Esta sentencia nos permite obtener el tamaño del arreglo.

Recuerde:

Cuando los arreglos no son definidos como parámetros estos deben inicializarse.

En resumen se puede declarar el arreglo y luego inicializarlo de las siguientes formas:

La primera:

double[] a ; declara sin inicializar el arreglo
a = new double[10]; inicializa 10 espacio de memoria

La segunda:

int double[] a={1,2,3,4,5,6,7,8,9,10}; declara e inicializa



Ahora busque los ejercicios del texto básico en la sección 8.1.1 “Ejercicios resueltos para unidimensionales” del texto básico, y encuentre cuales se declaran de la primera forma y cuales de la segunda.

En lo que respecta a **arreglos bidimensionales** y la forma de declaración en los algoritmos usted lo encontrará en la sección 8.2 “Arreglos bidimensionales” del texto básico. Se puede observar que simplemente se agrega un par de corchetes “[] [] ” en su declaración y por el resto es similar a la declaración de los arreglos unidimensionales.

A continuación se presenta algunos ejemplos en java donde existen declaración de arreglos bidimensionales:

```
11
12 public static void main(String[] args) throws IOException{
13     Scanner lector= new Scanner(System.in);
14     int M,N;
15     int mayor, menor;
16     int[][] Matriz;
17
18     System.out.println("Introduce el orden de la matriz MXN");
19     System.out.print("Introduce el número de filas M:");
20     M=lector.nextInt(); //Ingreso por teclado del día
21     System.out.print("Introduce el número de columnas N:");
22     N=lector.nextInt(); //Ingreso por teclado del mes
23
24
25     Matriz = new int [M][N];
```

Ilustración 3 Declaración e inicializar de una matriz MXN

Igual que en los arreglos unidimensionales, con base en la Ilustración 58, los arreglos se pueden definir como en la línea 16 y se los inicializa como en la línea 25. También se lo puede hacer en una sola línea `int [][] Matriz = new int[M][N];`. Observe que los valores **N** y **M** son ingresados por teclado lo que hace que el arreglo se defina dinámicamente por el usuario.




```

105 static void presentar(int[][] presentar){
106     System.out.printf("=====\n");
107     for (int i=0;i<presentar.length;i++) {
108         for (int j=0;j<presentar.length;j++){
109             System.out.printf("  %d",presentar[i][j]);
110         }
111     System.out.printf("\n");
112 }
113
114
115 }
116 }

```

Ilustración 4 Utilización de la sentencia *length*

En la Ilustración 59 observe como se utiliza la palabra reservada **length** para determinar el número de filas y columnas (línea 107 de la Ilustración 59) . Por lo general cuando se recorren arreglos bidimensionales siempre existirán dos ciclos FOR, uno que recorre filas y otro ciclo anidado que recorre columnas.

```

15
16 public static void main (String[] args ) throws IOException {
17
18     int Matriz [][]={{1,2,3,4},{-1,-2,-3,-4}};
19     int filas= Matriz.length;
20     int columnas=Matriz[0].length;
21     System.out.println("Corresponde al número de Filas "+filas);
22     System.out.println("Corresponde al número de columnas "+columnas);
23
24 }

```

Ilustración 5. Inicialización de una matriz con valores por defecto

En la línea 18 de la Ilustración 60, se declara una matriz junto con su inicialización ubicando valores ya en el arreglo. Observe como se distribuyen las llaves para dividir las filas y las columnas. En este programa también se puede recalcar como se obtienen las filas y columnas del arreglo con **length**. Esto es muy útil cuando se desconoce de las dimensiones de columna y fila. Observe la corrida del programa en la siguiente imagen:

```

run:
Corresponde al número de Filas 2
Corresponde al número de columnas 4
BUILD SUCCESSFUL (total time: 1 second)

```

Ilustración 6 Log de la ejecución de la Ilustración 60 mostrando la utilidad de *length*

Ahora repase los ejercicios de la sección 8.2.1 "Ejercicios resueltos para bidimensionales" del texto básico y trate de declarar los arreglos que se



presentan en Java, eso le permitirá tener una mayor destreza en declarar arreglos bidimensionales.

Finalmente revise las secciones 8.3 “Arreglos tridimensionales” y 8.4 “Arreglos tetradimensionales” del texto básico, para entender como se declaran los arreglos tridimensionales y tetradimensionales. De manera personal, podemos comentar que existen pocas aplicaciones donde se utilicen, aunque, en la sección 8.3.1 “Ejercicios resueltos para tridimensionales” y 8.4.1 “Ejercicios resueltos para tetradimensionales”, del texto básico se observa algunas utilidades. La Ilustración 62 muestra un ejemplo de la declaración de estos arreglos en Java:

```
16 public static void main (String[] args ) throws IOException {
17
18     int Matriz [][][]={
19         {{1,2,3,4},{-1,-2,-3,-4}},
20         {{1,2,3,4},{-1,-2,-3,-4}},
21         {{1,2,3,4},{-1,-2,-3,-4}}
22     };
23     int dim1= Matriz.length;
24     int dim2=Matriz[0].length;
25     int dim3=Matriz[0][0].length;
26     System.out.println("Corresponde al Primer dimensión:"+dim1);
27     System.out.println("Corresponde a la segunda dimensión "+dim2);
28     System.out.println("Corresponde a la tercera dimensión "+dim3);
29
30 }
31
32
33 }
```

27:56 INS

Output – Guia (run)

run:
Corresponde al Primer dimensión:3
Corresponde a la segunda dimensión 2
Corresponde a la tercera dimensión 4
BUILD SUCCESSFUL (total time: 1 second)

Ilustración 7 Arreglos tridimensionales

Una vez hechas las declaraciones de arreglos podríamos recorrer el arreglo para ingresar datos en cada una de sus posiciones o para presentar los valores que tenga asignados.



4.3 Recorrido / inserción de datos



Provéase de su texto básico y lea detenidamente las páginas: 166, 174, 185 y 190. Analice detenidamente los ejercicios que ahí se proponen e identifique la forma como se realizan las actividades de recorrido e inserción en un arreglo de cualquier dimensión.

¿Qué le pareció la lectura? ¿Comprendió? Si es así avancemos y repasemos ciertos elementos de la lectura anterior.

Si usted ha entendido la definición y la declaración de los arreglos ahora podrá realizar los recorridos y la inserción de datos. Cada posición del arreglo ya sea unidimensional o bidimensional se considera como una variable, entonces podemos asignar datos o valores con el nombre de la variable y el índice (ó índices) del arreglo.

Para comprender mejor la manipulación de los datos en los arreglos lea con atención los siguientes apartados del texto básico:

- Sección 8.1 Tema: *Manejo de los elementos del arreglo unidimensional.*
- Sección 8.2 Tema: *Manejo de los elementos del arreglo bidimensional.*
- Sección 8.3 Tema: *Manejo de los elementos del arreglo tridimensional.*
- Sección 8.4 Tema: *Manejo de los elementos del arreglo tetradimensional.*

En estas secciones se encuentran ejemplos donde se puede ver claramente que la inserción y el recorrido de los datos se los realiza mediante ciclos FOR y se puede decir que dependiendo de la dimensión del arreglo se declara un conjunto de FOR anidados.

La Ilustración 63 muestra la representación en Java del algoritmo del ejercicio 8.1.1.3 desarrollado en el texto básico, en donde el ingreso de los datos por teclado se realiza para el vector **a** en las líneas 17 – 20 y para el vector **b** en las líneas 21 – 24. También observe en las líneas 27 – 31 como se recorren los arreglos para realizar la multiplicación de los arreglos.



```

8 public class productovectores {
9     public static void main(String[] args) {
10         //Declaración de variables
11         int vectorA[]=new int[10];
12         int vectorB[]=new int[10];
13         int r,producto;
14         //leer desde teclado el valor de los vectores
15         Scanner lector= new Scanner(System.in);
16
17         for (int i=0; i<=9; i++){
18             System.out.printf("Ingrese el el valor del VectorA[%d]",i);
19             vectorA[i]=lector.nextInt(); //toma de datos desde el teclado
20         }
21         for (int i=0; i<=9; i++){
22             System.out.printf("Ingrese el el valor del VectorB[%d]",i);
23             vectorB[i]= lector.nextInt(); //toma de datos desde el teclado
24         }
25         System.out.println("Realizando operacion de producto vector");
26         producto=0;
27         for (int i=0; i<=9; i++){
28             System.out.printf("Producto del valor del VectorA %d * el valor " +
29                             "del VectorB %d\n",vectorA[i],vectorB[i]);
30             producto=producto + vectorA[i]*vectorB[i];
31         }
32         System.out.println("=====");
33         System.out.printf("El resultado de el producto del vector" +
34                             " A * Vector B es %d\n ",producto);
35     }
36 }

```

Ilustración 8 Programa en java para calular el producto de dos vectores unidimensionales AXB

En la ilustración siguiente observe un ejemplo en Java de un arreglo bidimensional. Realice un análisis del ejercicio 8.2.1.3 del texto básico para

entender el programa. Luego identifique en el programa el ingreso de datos y el recorrido para las operaciones solicitadas en el problema.



```

6 //realizado por greyson
7 import java.util.*;
8 public class ProduccionObreros {
9     public static void main(String[] args) {
10         Scanner lector = new Scanner(System.in);
11         //Declaración de variables
12         String[] obreros = new String[20];
13         int[][] producción = new int[20][6];
14         int ren, col, totProd, toTotProd;
15         //leer desde teclado el valor de los vectores
16         for (ren = 0; ren <= 19; ren++) {
17             System.out.printf("Ingrese el Nombre del Obrero[%d]",ren);
18             obreros[ren] = lector.next(); //toma de datos desde el teclado
19             for (col = 0; col <= 5; col++) {
20                 System.out.printf("Ingrese la producción de mes [%d] del Obrero [%d]"
21                                     ,col, ren);
22                 producción[ren][col] = lector.nextInt();
23             }
24         }
25         toTotProd = 0;
26         for (ren = 0; ren <= 19; ren++) {
27             System.out.printf("La producción del Obrero [%s]:",obrero[ren]);
28             totProd = 0;
29             for (col = 0; col <= 5; col++) {
30                 System.out.printf(" Mes [%d]: %d ",col,producción[ren][col]);
31                 totProd = totProd + producción[ren][col];
32             }
33             System.out.printf(" Total: %d \n",totProd);
34             toTotProd = toTotProd + totProd;
35         }
36         System.out.println("=====");
37         System.out.printf("El resultado total de la producción es %d \n", toTotProd);
38     }
}

```

Ilustración 9 Programa en Java ejercicio 8.2.1.3

Una vez que conocemos las operaciones básicas para el trabajo con arreglos, veamos, a través de ejercicios, algunas aplicaciones prácticas de los arreglos.

4.4 Ejercicios

Importante: en el análisis de los ejercicios desarrollados se utiliza números para referirse a líneas de código de los programas Java. Mientras que, para los algoritmos escritos en pseudocódigo se utiliza letras del alfabeto y a veces la combinación de letras y números.

1. En una matriz de dimensiones $m \times n$ hallar el mayor valor y el menor valor del arreglo.

Algoritmo ENCONTRAR MAYOR MENOR EN ARREGLO

Clase Arreglos.

1. Método Principal

a. Declaraciones



```
1. m,n, mayor,menor i, j: entero
2. Frase : cadena
3. Matriz :Arreglo[][] entero
b. Solicitar el orden de la matriz
c. Leer m,n
d. Inicializar Matriz [m][n]
e. Solicitar los datos de la matriz
f. Leer Matriz[m][n]
g. Mayor=0;
h. FOR i=0; i<m ; i++
    1. FOR j=0 ; j<n ; j++
        a. IF mayor<Matriz[i][j] THEN.
            1. Asignar mayor =Matriz[i][j]
        b. ENDIF
    2. ENDFOR
i. ENDFOR
j. Imprimir el mayor
k. menor = mayor
l. FOR i=0 ; i<m ; i++
    1. FOR j=0 ; j<n ; j++
        a. IF menor>Matriz[i][j] THEN
            1. menor =Matriz[i][j]
        b. ENDIF
    2. END FOR
m. ENDFOR
o. Imprimir el menor
2. Fin del método principal
Fin de la clase
```

Código en Java:



```

6 package Fundamentos;
7 import java.io.*;
8 import java.util.Scanner;
9 public class arreglos1 {
10     public static void main(String[] args) throws IOException{
11         Scanner lector= new Scanner(System.in);
12         int M,N;
13         int mayor, menor;
14         int[][] Matriz;
15         System.out.println("Introduce el orden de la matriz MXN");
16         System.out.print("Introduce el número de filas M:");
17         M=lector.nextInt(); //Ingreso por teclado del día
18         System.out.print("Introduce el número de columnas N:");
19         N=lector.nextInt(); //Ingreso por teclado del mes
20         Matriz = new int [M][N];
21         //Ingreso de valores en la matriz
22         System.out.printf("Introduce los datos en la matriz de %dX%d\n",M,N);
23         for (int i=0; i<M;i++)
24             for (int c=0;c<N;c++){
25                 System.out.printf("Valor de [%d,%d]:",i,c);
26                 Matriz[i][c]= lector.nextInt();
27             }
28         //procedimiento para encontrar el mayor
29         mayor=0;
30         for (int i=0;i<M;i++)
31             for (int c=0;c<N;c++)
32                 if (mayor<Matriz[i][c]) mayor =Matriz[i][c];
33         System.out.printf("El elemento mas grande es %d\n",mayor);
34         //procedimiento para encontrar el menor
35         menor=mayor;
36         for (int i=0;i<M;i++)
37             for (int c=0;c<N;c++)
38                 if (menor>Matriz[i][c]) menor=Matriz[i][c];
39         System.out.printf("El elemento mas pequeño es %d\n",menor);
40     }
41 }

```

Ilustración 10 Programa Java que busca el elemento menor y mayor de una matriz

Análisis

Es necesario recordar lo siguiente: para el análisis del código Java todas los números de líneas hacen referencia a la Ilustración 65.

El propósito del programa es encontrar dos elementos de una matriz que es ingresada por el usuario, cuyos valores represente el elemento mayor y el menor.

El proceso inicia con la definición de las variables a utilizar representadas en las líneas 12 a 14. Observe que en la línea 14 solo define la variable de la matriz y en la 20 se instancia con valores de **M** (filas) y **N** (columnas). Estos valores son ingresados en tiempo de ejecución en las líneas 17 y 19.

Seguidamente se observa el proceso de ingreso de la matriz. Esto se realiza mediante dos estructuras FOR anidadas cuyos valores finales son **M** y **N**. También los índices **i**, **c** representan cada uno de los valores de los elementos a ingresar en la línea 26.



Para encontrar el valor mayor, el proceso inicia asignando 0 a la variable

mayor. Luego con dos FOR anidados se recorre la matriz preguntando (línea 32) si cada uno de los elementos **M[i,c]** es mayor que la variable **mayor**, si esto es afirmativo se reemplaza el contenido de la variable por el nuevo elemento mayor. De esta forma se asegura el valor mayor en la variable.

El proceso para encontrar el valor menor es el mismo que para el mayor con la diferencia que se trabaja con la variable **menor** (línea 35).

Finalmente se imprime los valores.

2. Construya un algoritmo que simule un juego, que consisten en ingresar 3 números y determinar cuantos coinciden con 3 números generados aleatoriamente por el computador.

Algoritmo JUEGO ALEATORIO

Clase Arreglos2.

1. Método Principal

a. Declaraciones

1. m, n, i, j, aciertos: entero

2. rand : arreglo[][] entero

b. aciertos =0, m=2, n=3

c. Inicializar rand[m][n]

d. FOR i=0 ; i<N ; i++

1. Rand[0][i]= NumeroAleatorio (hasta 10)

2. Leer elemento rand [1][i]

e. ENDFOR

f. FOR i=0 ; i<3 ; i++

1. FOR j=0 ; j<3 ; j++

a. IF rand[1][i]==rand[0][j] THEN

1. aciertos =aciertos +1

b. ENDIF

2. ENDFOR

g. ENDFOR

h. IF aciertos=0 THEN

1. Imprimir no hay aciertos

i. ELSE

1. Imprimir existes aciertos

j. ENDIF

2. Fin del método principal

Fin de la clase

Código en Java:




```

9  import java.util.*;
10 public class arreglos2 {
11
12     public static void main(String[] args) {
13         Scanner lector= new Scanner(System.in);
14         Random aleatorios = new Random();
15         int M=2,N=3;
16         int [][]rand=new int[M][N];
17         int aciertos=0;
18         for(int i=0; i<N;i++){
19             rand[0][i]=aleatorios.nextInt(10);
20             System.out.printf("Introduce un numero:%d\n ",rand[0][i] );
21             rand[1][i] =lector.nextInt();
22         }
23
24         System.out.println("-----");
25         for(int i=0; i<3;i++){
26             for(int j=0; j<3;j++){
27                 if(rand[1][i]==rand[0][j]){
28                     aciertos = aciertos + 1;
29                 }
30             }
31         }
32         if(aciertos==0)
33             System.out.println("No has acertado ninguno");
34         else
35             System.out.printf("Has acertado %d veces\n",aciertos);
36     }
37 }

```

Ilustración 11 Programa Java que cuenta los aciertos

Análisis

Cabe señalar que: para el análisis del código Java todas los números de líneas hacen referencia a la Ilustración 66.

El objetivo del programa es conocer cuantos aciertos tiene un usuario al ingresar tres números y compararlos con tres números que el computador genera aleatoriamente. En la línea 14 se define un objeto de la clase **Random** que nos permite generar números aleatorios mediante el procedimiento **nextInt()** tal como se lo realiza en la línea 19.

Se inicia con la definición de las variables **M** y **N** tendrán el valor de 2, 3 respectivamente. Los valores corresponden al número de filas y el de columnas. Los aleatorios serán almacenados en la primera fila y los ingresados por el usuario en la segunda (por esto **M=2**, **N=3**). En la línea 18 inicia una estructura FOR que permitirá realizar dos tareas: la primera corresponde a generar un numero aleatorio de 0 a 9 y almacenar el valor en la matriz **random[0][i]**; la segunda corresponde a almacenar el valor que ingrese el usuario mediante **lector.nextInt()** en **rand[1][i]**.

El proceso para contar los aciertos, inicia en la línea 24 con dos estructuras FOR anidadas. En la línea 27, se evalúa si algún elemento de la primera fila (**rand[0][j]**) es igual a algún elemento de la segunda fila **rand[1][i]**. La línea 28 contabilizará la cantidad de aciertos.



Finalmente presentamos el resultado, teniendo en cuenta que si *aciertos* es igual a cero, presentaremos un mensaje indicando que no se ha acertado y caso contrario las veces que ha acertado.

3. Crear una matriz de 5X5 en donde cada elemento de ésta corresponde a la suma de los índices de la fila con la columna (i+j).

Algoritmo SUMAR SUBINDICES DE MATRIZ

Clase Arreglos3.

1. Método Principal

a. Declaraciones

1. m,n i, j: entero

2. matriz : arreglo[][] entero

b. Solicitar dimensiones m, n

c. Leer m,n

d. Inicializar valor matriz[m][n]

e. FOR i=0 ; i<M ; i++

1. FOR j=0 ; j<N ; j++

a. Asignar Matriz[i][j]= i+j

2. ENDFOR

f. ENDFOR

g. Imprimir encabezado

h. FOR i=0 ; i<M ; i++

1. FOR j=0 ; j<N ; j++

1. Imprimir Matriz[i][j]

2. ENDFOR

i. ENDFOR

2. Fin del método principal

Fin de la clase

Código en Java:



```

8  import java.util.Scanner;
9  /**
10   * @author greyson
11   */
12  public class arreglos3 {
13      public static void main(String[] args){
14          Scanner lector= new Scanner(System.in);
15
16          int M,N;
17          int[][] Matriz;
18
19          System.out.println("Introduce el orden de la matriz MXN");
20          System.out.print("Introduce el número de filas M:");
21          M=lector.nextInt();
22          System.out.print("Introduce el número de columnas N:");
23          N=lector.nextInt();
24
25          Matriz = new int [M][N];
26          //Ingreso de valores en la matriz
27          for (int i=0; i<M;i++){
28              for (int j=0;j<N;j++){
29                  Matriz[i][j]= i+j;
30              }
31
32              //procedimiento para encontrar presentar la matriz
33              System.out.print("=====\n");
34              for (int i=0;i<M;i++) {
35                  for (int j=0;j<N;j++){
36                      System.out.print(" "+Matriz[i][j]);
37                  }
38                  System.out.println("\n");
39              }
40      }
41  }

```

Ilustración 12 Programa Java que suma los índices de una matriz

Análisis

No olvide que: para el análisis del código Java todas los números de líneas hacen referencia a la Ilustración 67.

El propósito del programa consiste en almacenar en cada elemento de una matriz M X N la sumatoria de los índices que identifican ese elemento. Por ejemplo el primer elemento será cero, porque el índice de fila $i=0$ y columna $j=0$ entonces $\text{Matriz}[i][j] = 0 + 0 (i,j) = 0$. Al igual que en ejercicios anteriores se solicita al usuario que ingrese M y N (líneas 21 y 23) para inicializar la variable Matriz (línea 25).

A continuación una estructura FOR anidada se utiliza para recorrer la matriz y en la línea 29, se realiza la operación de asignar la suma de índices al elemento tal como el ejemplo anterior.

Finalmente en la línea 34 inicia otra estructura FOR anidada que



permitirá visualizar los datos guardados en los elementos.

4. Se pide diseñar e implementar un algoritmo que permita multiplicar dos matrices A y B. Además, se debe validar el orden de cada una de las matrices para realizar la operación.

Algoritmo MULTIPLICAR MATRICES

Clase Arreglos4.

1. Método Principal

a. Declaraciones

1. ma,na,mb,nb: i,j, suma : entero
2. a , b, producto : arreglo[][] entero

b. suma=0

c. DO

1. Solicitar orden de arreglo a
2. Leer ma,na
3. Solicitar orden de arreglo b
4. Leer mb,na
5. Solicitar la frase

d. WHILE na=mb

e. Inicializar arreglo a[ma][na]

f. Inicializar arreglo b[mb][nb]

g. Inicializar producto a[ma][nb]

h. Solicitar Arreglo a

i. Leer a[][]

j. Solicitar arreglo

k. Leer b[][]

m. FOR i = 0 ; i < ma ; i++

1. FOR j = 0 ; j < nb ; j++)

a. suma=0

b. FOR k = 0 ; k < mb ; k++

1. suma += A[i][k] * B[k][j];

c. ENDFOR

d. producto[i][j] = suma;

2. ENDFOR

n. ENDFOR

o. Imprimir el arreglo producto[][];

2. Fin del método principal

Fin de la clase

Código en Java:



```

8 import java.util.*;
9 public class arreglos4 {
10     public static void main(String[] args){
11         Scanner lector= new Scanner(System.in);
12         int M_A,N_A,M_B,N_B;
13         int[][] A;
14         int[][] B;
15         int[][] producto;
16
17         int suma = 0;
18         //Ingreso deL orden de la matriz
19         do {
20             System.out.println("Introduce el orden de la matriz A MXN");
21             System.out.print("Introduce el número de filas M:");
22             M_A=lector.nextInt(); //Ingreso por teclado del día
23             System.out.print("Introduce el número de columnas N:");
24             N_A= lector.nextInt();
25
26             System.out.println("Introduce el orden de la matriz B MXN");
27             System.out.print("Introduce el número de filas M:");
28             M_B=lector.nextInt();
29             System.out.print("Introduce el número de columnas N:");
30             N_B=lector.nextInt();
31             if (N_A!=M_B){
32                 System.out.println("El orden no es válido para realizar la" +
33                                     " multiplicación:");
34             }
35         }while(N_A!=M_B);
36         A = new int [M_A][N_A];
37         B = new int[M_B][N_B];
38         producto= new int [M_A][N_B];
39         //Ingreso de número aleatoriamente
40         Random aleatorios = new Random();
41
42         for(int i=0; i<M_A;i++)
43             for (int j=0;j<N_A;j++)
44                 A[i][j]=aleatorios.nextInt(10);
45
46         for(int i=0; i<M_B;i++)
47             for (int j=0;j<N_B;j++)
48                 B[i][j]=aleatorios.nextInt(10);
49
50         for(int i = 0; i < M_A; i++){
51             for(int j = 0; j < N_B; j++){
52                 suma = 0;
53                 for(int k = 0; k < M_B; k++){
54                     suma += A[i][k] * B[k][j];
55                 }
56                 producto[i][j] = suma;
57             }
58         }
59
60         //procedimiento para encontrar presentar la matriz
61         System.out.print("=====\\n");
62         for (int i=0;i<M_A;i++) {
63             for (int j=0;j<N_B;j++){
64                 System.out.print(" "+producto[i][j]);
65             }
66             System.out.println("\\n");
67         }
68     }
69 }

```

Ilustración 13 Programa en Java que multiplica dos matrices



Análisis

Tenga presente que: para el análisis del código Java todas los números de líneas hacen referencia a la Ilustración 68.

El propósito del programa consiste en ingresar dos matrices, evaluar su orden y realizar la multiplicación de éstas en una nueva matriz.

Se utilizará **M_A** para representar la fila y **N_A** para la columna de la primera matriz; **M_B** para la fila y **N_B** para la columna en la segunda matriz. Además se define la matriz **A**, **B** y **producto** para realizar la operación.

Desde la línea 19 hasta la 35, se realiza la validación de los ordenes de las matrices **A** y **B**.

En la línea 31 se verifica si **N_A** es igual a **M_B** como requisito para poder realizar la multiplicación. En caso de que no sean iguales estas dos variables en la línea 35 realizamos la repetición hacia la línea 19 para que nuevamente se ingresen los valores.

En la línea 36 y 37 inicializamos las matrices **A**, **B** y **producto**. Observe que **producto** tendrá el orden **M_B** para la fila y **N_B** para la columna.

En este ejercicio no realizaremos el ingreso de los elementos de la matrices por teclado, simplemente se los genera mediante la clase **Random** como muestra la línea 40. A continuación observamos una estructura FOR anidada para el ingreso de los elementos en las matrices **A** y **B** (líneas 42 a la 48).

Como siguiente paso, en las líneas 50 a 58 se realiza la operación de multiplicación de las dos matrices. Le recomiendo que revise el proceso manual de multiplicación de matrices, para que entienda el porque los índices están distribuidos de la manera que muestra la línea 54.

Finalmente mediante dos FOR anidados se presenta el resultado de la multiplicación almacenada en **producto**.

5. Escriba un programa en pseudocódigo y en Java que lea una frase y muestre los contadores del número de tildes dentro de una frase.

Algoritmo CONTAR TILDES

Clase Arreglos9.

1. Método Principal

a. Declaraciones

1. tilde: arreglo[5] carácter



```

2. numTildes: arreglo[5] entero
3. contador, i, j : entero
4. frase: cadena
b. tilde={'á','é','í','ó','ú'}
c. Solicitar la frase
d. Leer frase
e. FOR i=0 ; i< 5; i++
    1. Asignar contador=0
    2. FOR j=0 ; j<longitud(frase) ; j++
        a. IF frase.subCaracter(j)==tilde[i] THEN
            1. contador=contador+1
        b. ENDIF
    3. ENDFOR
    4. numTildes[i]=contador
f. ENDFOR
g. FOR i=0 ; i< 5; i++
    1. Imprimir La Tilde[i] esta en la frase numTildes[i]
h. ENDFOR
2. Fin del método principal
Fin de la clase

```

Código en Java:

```

6 package Fundamentos;
7 import java.util.Scanner;
8 /* * @author greyson
9 */
10 public class arreglos9 {
11     public static void main(String[] args) {
12         Scanner lector= new Scanner(System.in);
13         char[] tilde={'á','é','í','ó','ú'};
14         int[] numTildes=new int[5];
15         int contador;
16         String frase;
17
18         System.out.print("Ingrese la frase u oración:");
19
20         frase= lector.nextLine();//Toma de datos desde el teclado
21         frase= frase.trim(); //Elimina los espacios a cada lado del texto
22         frase=frase.toLowerCase(); //Transforma la cadena a minúsculas
23
24
25         for(int i=0;i<5;i++){
26             contador=0;
27             for (int j=0;j<frase.length();j++)
28                 if(frase.charAt(j)==tilde[i]) contador++;
29             numTildes[i]=contador;
30         }
31
32
33         for(int i=0;i<5;i++)
34             System.out.printf("La tilde %s en la frase esta %d veces.\n",
35                             tilde[i],numTildes[i]);
36     }
37 }

```

Ilustración 14 Programa en Java que cuenta tildes



Análisis:

Recuerde que: para el análisis del código Java todas los números de líneas hacen referencia a la Ilustración 69.

El presente programa tiene la finalidad de contar cuantas tildes existen en una oración. El programa recibe como entrada una frase.

Observemos que se define un arreglo unidimensional de tipo **caracter** (línea 13) con elementos de cada una de las tildes . También definimos un arreglo unidimensional entero para ubicar el número de tildes por cada una de las vocales (línea 14).

Seguidamente solicitamos al usuario que ingrese la frase (línea 20), luego procedemos a borrarle los espacios a los extremos y a convertir toda la frase a minúscula con el fin de se puedan comparar con el arreglo tilde.

Luego, utilizamos una estructura de dos FOR anidados para recorrer los cada una de las letras de la frase y compararlos con cada elemento del arreglo tilde (Línea 28). Además el contador nos permitirá agregar una unidad(línea 29) si la comparación es verdadera. Al termino del FOR en **j** asignamos el valor de contador a **numTildes** (línea29) para tener registrado el número de tildes de cada una de las vocales.

Finalmente recorreremos **tilde** y **numTilde** presentando cada uno de los valores correspondientes a las vocales con tilde.

En este capítulo revisamos como definir y declarar arreglos de diferentes dimensiones, analizamos los diferentes conceptos de la declaración de arreglos y como se puede hacer la inserción y el recorrido de los datos. Finalmente se ha visto algunos ejercicios en Java como traducción de algunos algoritmos del texto básico.

Para ampliar los conocimientos sobre arreglos les proponemos realizar la siguiente lectura que les ayudará a comprender de mejor manera los temas aquí explicados y les presentará otros contenidos relacionados.

Material adicional: En el CD que se adjuntó en la guía didáctica, puede revisar (programación-en-java-i\Contenidos\LecturaObligatoria\) el archivo: 11-algunasclasesestandardejavaii



Para medir el nivel de comprensión del tema le invitamos a que desarrolle la siguiente autoevaluación. Trate de resolver usted mismo los problemas, para luego revisar y comparar con las soluciones que proponemos al final de la



Esta obra ha sido licenciada con Creative Commons Ecuador 3.0 de Reconocimiento - No comercial - Compartir igual (<http://creativecommons.org/licenses/by-nc-sa/3.0/ec/>).

presente guía didáctica.

