



2022.

Grupo PATC

Tema a tratar:

Arreglos en JAVA

CURN

Integrantes:

Angel David Ariz Lambraño
Luis Miguel Uparela Cuellar
Juan David Fuentes Miranda
Juan David Duarte Mendoza
David Joé Patiño Leal.

Introducción.

Somos estudiantes de la Corporación Universitaria Rafael Nuñez, por medio de este trabajo estaremos dando a conocer nuestro conocimiento sobre los Arreglos, de acuerdo a nuestro conocimiento, todo lo investigado y todo lo aprendido. Estarán viendo los concepto de cada uno de los tipos de ciclos, anexados de un ejemplo y un video explicativo el cual servirá de guía para un mejor aprendizaje.

Tabla de Contenido Preparación del Documento.

¿Cómo declarar arreglos estáticos de una dimensión?.....	2
¿Cómo llenar un vector?.....	3
¿Cómo acceder a un elemento de un vector?.....	4
Arreglos multidimensionales o matrices.....	5
¿Cómo acceder a los elementos de una matriz?.....	7
Ejemplo completo de una matriz.....	8

¿Cómo declarar arreglos estáticos de una dimensión?

Para declarar arreglos de una dimensión o arreglos unidimensionales, utilizaremos la siguiente sintaxis:

<pre>Tipo_dato[] Nombre del arreglo = new Tipo_dato[tamaño o dimensión]; int[] Arreglo = new int[6]; //Ejemplo de declaración</pre>
--

int es el tipo de dato para los elementos del vector, si lo declaramos así el vector solo puede almacenar datos de tipo entero.

[] Los corchetes le indican al compilador que estamos declarando un vector, más no una variable.

“Arreglo” es el nombre que se le da al vector, con este nombre podré acceder al contenido del vector las veces que sea necesario.

= Este operador le dice al compilador que se debe dirigir a una zona de memoria porque se va a reservar algo para el arreglo.

La palabra reservada “new” establece un espacio en la memoria para el vector, y la palabra int le dice que será de tipo entero.

[6] El número dentro de los corchetes le indica que el tamaño o la dimensión de ese arreglo será de. Lo que quiere decir que solo se pueden almacenar 6 elementos de tipo entero. Cabe resaltar que el tamaño o dimensión de un arreglo es fija; Y no se puede cambiar en tiempo de ejecución.

Los tipos de datos pueden ser cualquiera de los admitidos por JAVA . Unos ejemplos de declaración e inicialización con valores por defecto de arreglos utilizando todos los tipo de datos en JAVA, son:

```
byte[ ] edad = new byte [4];
short[ ] edad = new short [4];
int[ ] edad = new int [4];
long[ ] edad = new long [4];
float[ ] estatura = new float[3];
double[ ] estatura = new double[3];
boolean[ ] estado = new boolean[5];
char[ ] sexo = new char[2];
String [ ] nombre = new String[2];
```

La declaración del anterior vector en memoria se vería más o menos así.

Arreglo[0]	Arreglo[1]	Arreglo[2]	Arreglo[3]	Arreglo[4]	Arreglo[5]
0	1	2	3	4	5

¿Cómo llenar un vector?

Un vector se maneja a través de posiciones, iniciando en la posición 0, y para obtener o almacenar uno de sus elementos se usa una variable. La variable indica la posición del elemento en el vector a la que se quiere acceder.

Entonces un vector con tamaño o dimensión 6 se pueden almacenar 6 elementos pero solo tendrá 5 posiciones ya que la primera posición es 0. A estas posiciones se puede acceder ya sea para guardar u obtener elementos almacenados o guardados.

Para guardar un elemento en un vector se utiliza esta sintaxis:

Nombre_vector[posición] = valor_digitado o asignado;

Vamos a utilizar el ejemplo anterior para dar una muestra:

```
Arreglo[0] = 1;
```

Lo que se hizo fue asignarle el valor 1 a la posición 0 del vector.

Otra manera de iniciar un vector, es dar sus elementos al declararlo. Si lo hacemos de esta manera, no tendríamos que declarar un tamaño o una dimensión, ya que el compilador asigna la dimensión de acuerdo a los elementos con los que se inicializa el vector.

```
Tipo_dato [ ] nombre_variable = {valor1, valor2, valor3, ...};
```

Un ejemplo de la exterior sintaxis sería:

```
int [ ] Arreglo = {0, 1, 2, 3, 4, 5};
```

¿Cómo acceder a un elemento de un vector?

Para obtener un elemento de un vector, se realiza de esta manera :

```
Tipo_dato nombre_variable = nombre_vector[posición];  
int valor = Arreglo[2];
```

Se está obteniendo el elemento de la posición 2 y se almacena en una variable de tipo entero llamada valor .

Cuando el tamaño o la dimensión de un arreglo es demasiado grande, se puede llenar o imprimir el contenido de un vector con un ciclo for.

Un ejemplo completo de todo esto será explicado en el siguiente video:

Los valores por defecto son :

Para números es el valor "0".

Para cadenas y letras el valor "vacío".

Para booleanos el valor "falso".

Si queremos iniciar con valores propios, lo que se hará será esto:

Para números enteros:

```
int [ ] edad = {10, 87, 65, 11};
```

De la misma manera se haría para los otros tipos de enteros los cuales son: byte, short, long.

Para números reales:

```
double [ ] estatura = {1.82, 1.78, 1.69};
```

De igual forma lo haríamos con el tipo float, pero teniendo en cuenta que los números deben llevar una "f" al final, como por ejemplo, 1.82f o 1.82F.

Para cadenas:

```
String [ ] nombre = {"Angel", "David"};
```

Para caracteres:

```
char [ ] sexo = {'m', 'f'};
```

Para booleanos:

```
boolean [ ] = {true, false};
```

Arreglos multidimensionales o matrices.

Los arreglos multidimensionales también son conocidos como matrices, Estas funcionan de forma semejante a un vector, ya que como un vector estas matrices también almacenan elementos de un mismo tipo, el acceso a sus elementos se hace usando posiciones igual que en los vectores.

La diferencia está en que en las matrices se utiliza el concepto de fila y columna y debido a esto se trabajan de forma distinta.

¿Cómo se declara una matriz?

La declaración de una matriz se puede hacer con las siguiente sintaxis:

```
Tipo_dato nombre_matriz [ ] [ ] = new Tipo_dato [dimensión_filas] [dimensión_columnas];  
  
int Matriz [ ] [ ] = new int[3] [3]; //ejemplo de inicialización de la matriz
```

int es el tipo de dato para los elementos de la matriz, lo que quiere decir que la matriz solo podrá tener elementos de tipo entero.

[] [] Los dos corchetes le indican al compilador que estoy declarando una matriz, no una variable ni un vector.

“Matriz” es el nombre que se le da a la matriz, con este nombre podré acceder al contenido de la matriz las veces que sea necesario.

= Este operador le dice al compilador que se debe dirigir a una zona de memoria porque se va a reservar algo para la matriz.

La palabra reservada “new” establece un espacio en la memoria para la matriz, y la palabra int le dice que será de tipo entero.

[3] [3] el número que está entre corchetes le indica que la dimensión que está en la matriz, el primer número entre corchetes dice cuántas filas tendrá la matriz y el segundo corchete indica cuántas columnas tendrá la matriz , es decir tendrá 3 filas y 3 columnas.

Existen 2 maneras de llenar una matriz.

La primera es dando valores al momento de crearla, y la sintaxis sería:

```
Tipo_dato nombre_matriz [ ] [ ] = {{elemento1, elemento2, elemento3}, {elemento4, elemento5, elemento6}};  
int Matriz= {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
```

Acabamos de llenar una matriz con elementos fijos y de una dimensión 3x3. Las llaves externas definen la matriz y las filas se separan utilizando llaves internas seguridad de una coma (,). Para separar las columnas se pone una coma (,) después de cada elemento.

La otra manera de declarar una matriz es dando las dimensiones y luego llenar las posiciones teniendo en cuenta esta sintaxis

```

nombre_matriz[posfila][poscolumna]= elemento;
int miMatriz [][] =new int [3][3];
//primera fila
Matriz[0][0]=1;
Matriz[0][1]=2;
Matriz[0][2]=3;
//segunda fila
Matriz[1][0]=5;
Matriz[1][1]=6;
Matriz[1][2]=7;
//tercera fila
Matriz[2][0]=8;
Matriz[2][1]=9;
Matriz[2][2]=10;

```

La declaración en inicialización anterior establecerá en memoria algo así.

	Matriz[0]	Matriz[1]	Matriz[2]
Matriz[0]	1	2	3
Matriz[1]	4	5	6
Matriz[2]	7	8	9

Como podemos ver las matrices al igual que en los vectores inician desde la posición 0 , por eso para acceder o llenar un elemento, debemos tener en cuenta el no acceder a posiciones que no existen.

La manera anterior de como llenar una matriz será un poco complicada cuando nos toque llenar muchos elementos , entonces las líneas de código serían más extensas, a continuación veremos una forma más dinámica, mientras usamos 2 ciclos for , uno para las filas y otro para columnas.

```

Scanner sc = new Scanner(System.in);

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

System.out.print("Digite un elemento");

Matriz [ i ] [ j ] = sc.nextInt();
}
}

```

```
}
```

En este caso se han pedido los elementos en la consola, donde los índices o posiciones son las variables i e j, que son las que llenan la posición de la matriz.

¿Cómo acceder a los elementos de una matriz?

De la misma manera en la que se puede llenar una matriz usando las posiciones vistas anteriormente, también se puede acceder a sus elementos.

```
Tipo_dato variable = nombre_matriz[posfila][poscolumna]= elemento;  
  
int var= Matriz [0] [1];
```

Si continuamos con el ejemplo anterior el valor que se almacena en la variable “var” es el número 2 ya que está en la fila [0] y en la columna [1] de la matriz.

Otra manera de acceder a una matriz es utilizando un ciclo for para recorrerla, esta vez vamos a imprimir los elementos de la matriz anterior.

```
for(int i = 0; i < Matriz.length; i++){  
    for(int j = 0; j < Matriz.length; j++){  
        System.out.println("String format(\"\\n\", Matriz[ i ] [ j ]));  
    }  
}
```

Ejemplo completo de una matriz.

Digitar elementos de una matriz 3x3 y obtener e imprimir la suma y el promedio por cada fila de la matriz.

```
package ejemplos_arreglo;  
  
import java.util.Scanner;  
  
public class Ejemplos_Arreglo {  
  
    public static void main(String[] args) {  
        int Matriz[][] = new int[3][3];  
  
        Scanner entrada = new Scanner(System.in);
```

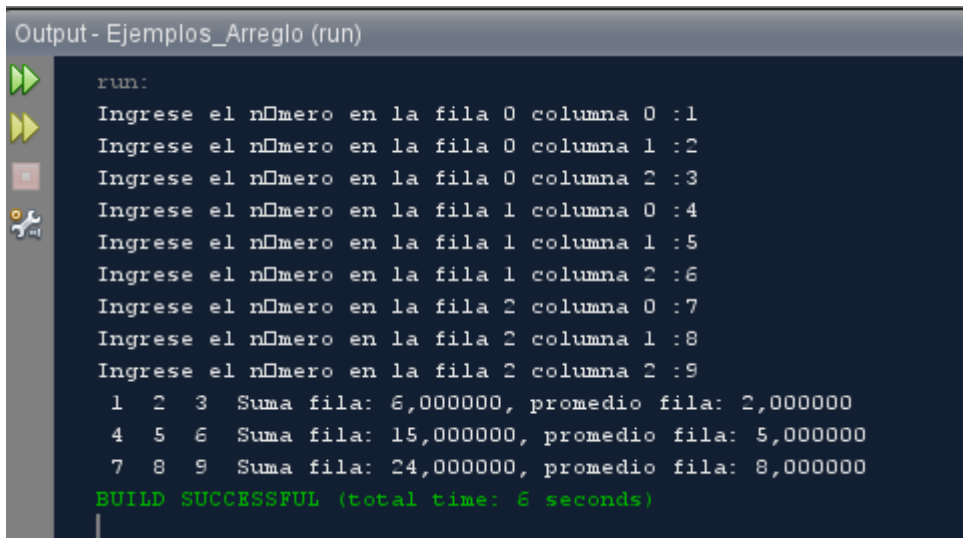


```

        float sumFila = 0;
        for (int i = 0; i < Matriz.length; i++) {
            for (int j = 0; j < Matriz.length; j++) {
                System.out.print("Ingrese el número en la fila " + (i)
                                + " columna " + j + " :");
                Matriz[i][j] = entrada.nextInt();
            }
        }
        for (int[] Matriz1 : Matriz) {
            sumFila = 0;
            for (int j = 0; j < Matriz1.length; j++) {
                sumFila += Matriz1[j];
                System.out.print(String.format(" %d ", Matriz1[j]));
            }
            System.out.print(String.format(
                " Sum fila: %f, promedio fila: %f ", sumFila, sumFila
                / Matriz.length));
            System.out.println();
        }
    }
}

```

La ejecución de este ejemplo sería más o menos así.



```

Output- Ejemplos_Arreglo (run)
run:
Ingrese el número en la fila 0 columna 0 :1
Ingrese el número en la fila 0 columna 1 :2
Ingrese el número en la fila 0 columna 2 :3
Ingrese el número en la fila 1 columna 0 :4
Ingrese el número en la fila 1 columna 1 :5
Ingrese el número en la fila 1 columna 2 :6
Ingrese el número en la fila 2 columna 0 :7
Ingrese el número en la fila 2 columna 1 :8
Ingrese el número en la fila 2 columna 2 :9
1 2 3 Suma fila: 6,000000, promedio fila: 2,000000
4 5 6 Suma fila: 15,000000, promedio fila: 5,000000
7 8 9 Suma fila: 24,000000, promedio fila: 8,000000
BUILD SUCCESSFUL (total time: 6 seconds)

```

Tanto en vectores o matrices los corchetes pueden ir delante o detrás del nombre y no existirían errores de compilación.

```

int [ ] vector = new int[10];
int vector [ ] = new int[10];

```

Por ultimo dejare un link de un video para que sea un poco más fácil comprender este tema:

<https://youtu.be/XEjdklYjh7g>