# 6.E. Arrays multidimensionales.

Sitio: Formación Profesional a Distancia
Curso: Programación
Libro: 6.E. Arrays multidimensionales.
Imprimido por: Iván Jiménez Utiel
Día: martes, 7 de enero de 2020, 22:35

## Tabla de contenidos

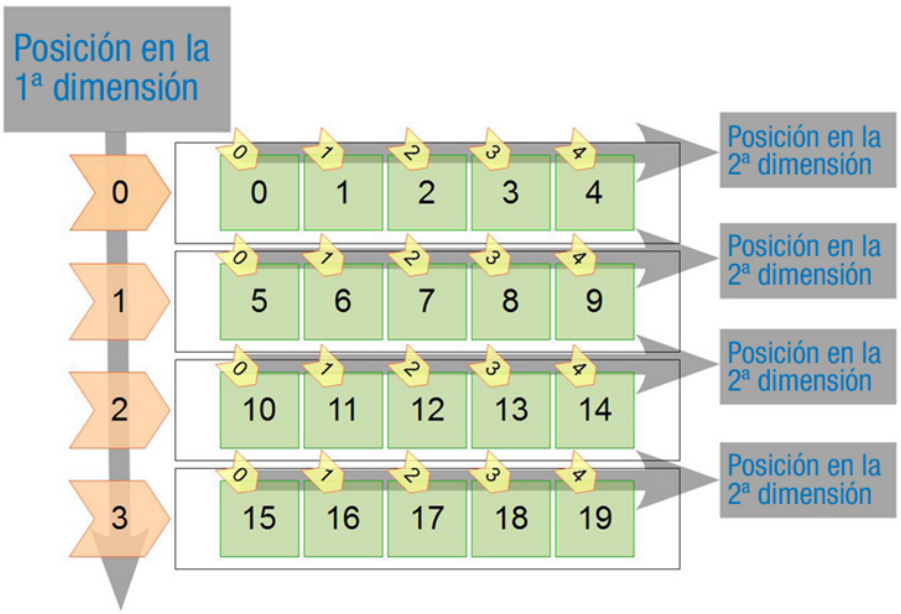
## 1. Arrays multidimensionales.

¿Qué estructura de datos utilizarías para almacenar los píxeles de una imagen digital? Normalmente las imágenes son cuadradas o rectangulares, así que una de las estructuras más adecuadas es la matriz. En la matriz cada valor podría ser el color de cada píxel. Pero, ¿qué es una matriz a nivel de programación? Pues es un array con dos dimensiones, o lo que es lo mismo, un array cuyos elementos son arrays de números.

Los arrays multidimensionales están en todos los lenguajes de programación actuales, y obviamente también en Java. La forma de crear un array de dos dimensiones en Java es la siguiente:

```
int[][] a2d=new int[4][5];
```

El código anterior creará un array de dos dimensiones, o lo que es lo mismo, creará un array que contendrá 4 arrays de 5 números cada uno. Veámoslo con un ejemplo gráfico:




Al igual que con los arrays de una sola dimensión, los arrays multidimensionales deben declararse y crearse. Podremos hacer arrays multidimensionales de todas las dimensiones que queramos y de cualquier tipo. En ellos todos los elementos del array serán del mismo tipo, como en el caso de los arrays de una sola dimensión. La declaración comenzará especificando el tipo o la clase de los elementos que forman el array, después pondremos tantos corchetes como dimensiones tenga el array y por último el nombre del array, por ejemplo:

```
int [][][] arrayde3dim;
```

La creación es igualmente usando el operador new, seguido del tipo y los corchetes, en los cuales se especifica el tamaño de cada dimensión:

```
arrayde3dim=new int[2][3][4];
```

Todo esto, como ya has visto en un ejemplo anterior, se puede escribir en una única sentencia.

**Autoevaluación**

**Completa con los números que faltan:**

```
int[][][] k=new int[10][11][12];
```

El array anterior es de _____ dimensiones, y tiene un total de _____ números enteros.

Resolver

### 1.1. Uso de arrays multidimensionales.

¿Y en que se diferencia el uso de un array multidimensional con respecto a uno de una única dimensión? Pues en muy poco la verdad. Continuaremos con el ejemplo del apartado anterior:

```
int[][] a2d=new int[4][5];
```

Para acceder a cada uno de los elementos del array anterior, habrá que indicar su posición en las dos dimensiones, teniendo en cuenta que los índices de cada una de las dimensiones empieza a numerarse en 0 y que la última posición es el tamaño de la dimensión en cuestión menos 1.

Puedes asignar un valor a una posición concreta dentro del array, indicando la posición en cada una de las dimensiones entre corchetes:

```
a2d[0][0]=3;
```

Y como es de imaginar, puedes usar un valor almacenado en una posición del array multidimensional simplemente poniendo el nombre del array y los índices del elemento al que deseas acceder entre corchetes, para cada una de las dimensiones del array. Por ejemplo:

```
int suma=a2d[0][0]+a2d[0][1]+a2d[0][2]+a2d[0][3]+a2d[0][4];
```

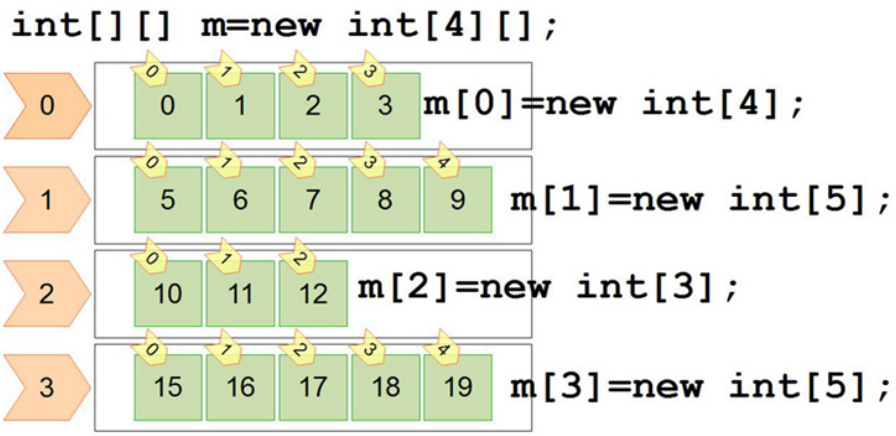
Como imaginarás, los arrays multidimensionales pueden también ser pasados como parámetros a los métodos, simplemente escribiendo la declaración del array en los argumentos del método, de forma similar a como se realizaba para arrays de una dimensión. Por ejemplo:

```
static int sumaarray2d(int[][] a2d) {
        int suma = 0;
        for (int i1 = 0; i1 < a2d.length; i1++)
            for (int i2 = 0; i2 < a2d[i1].length; i2++)  suma += a2d[i1][i2];
        return suma;
    }
```

Del código anterior, fijate especialmente en el uso del atributo `length` (que nos permite obtener el tamaño de un array). Aplicado directamente sobre el array nos permite saber el tamaño de la primera dimensión (`a2d.length`). Como los arrays multidimensionales son arrays que tienen como elementos arrays (excepto el último nivel que ya será del tipo concreto almacenado), para saber el tamaño de una dimensión superior tenemos que poner el o los índices entre corchetes seguidos de `length` (`a2d[i1].length`).

Para saber al tamaño de una segunda dimensión (dentro de una función por ejemplo) hay que hacerlo así y puede resultar un poco engorroso, pero gracias a esto podemos tener arrays multidimensionales irregulares.

Una matriz es un ejemplo de array multidimensional regular, ¿por qué? Pues porque es un array que contiene arrays de números todos del mismo tamaño. Cuando esto no es así, es decir, cuando los arrays de la segunda dimensión son de diferente tamaño entre sí, se puede decir que es un array multidimensional irregular. En Java se puede crear un array irregular de forma relativamente fácil, veamos un ejemplo para dos dimensiones.




- **Declaramos y creamos el array pero sin especificar la segunda dimensión.** Lo que estamos haciendo en realidad es crear simplemente un array que contendrá arrays, sin decir como son de grandes los arrays de la siguiente dimensión: `int[][] irregular=new int[3][];`

- **Después creamos cada uno de los arrays unidimensionales** (del tamaño que queramos) y lo asignamos a la posición correspondiente del array anterior: `irregular[0]=new int[7]; irregular[1]=new int[15]; irregular[2]=new int[9];`

**Recomendación**

Cuando uses arrays irregulares, por seguridad, es conveniente que verifiques siempre que el array no sea `null` en segundas dimensiones, y que la longitud sea la esperada antes de acceder a los datos:

```
if (irregular[1]!=null && irregular[1].length>10) {...}
```

## 1.2. Inicialización de arrays multidimensionales.

¿En qué se diferencia la inicialización de arrays unidimensionales de arrays multidimensionales? En muy poco. La inicialización de los arrays multidimensionales es igual que la de los arrays unidimensionales.

Para que una función retorne un array multidimensional, se hace igual que en arrays unidimensionales. Simplemente hay que poner el tipo de dato seguido del número de corchetes correspondiente, según el número de dimensiones del array. Eso claro, hay que ponerlo en la definición del método:

```
int[][] inicializarArray (int n, int m)
{
    int[][] ret=new int[n][m];
    for (int i=0;i<n;i++)
         for (int j=0;j<m;j++)
              ret[i][j]=n*m;
    return ret;
}
```

También se puede inicializar un array multidimensional usando las llaves, poniendo después de la declaración del array un símbolo de igual, y encerrado entre llaves los diferentes valores del array separados por comas, con la salvedad de que hay que poner unas llaves nuevas cada vez que haya que poner los datos de una nueva dimensión, lo mejor es verlo con un ejemplo:

```
int[][] a2d={{0,1,2},{3,4,5},{6,7,8},{9,10,11}};
int[][][] a3d={{{0,1},{2,3}},{{0,1},{2,3}}};
```

El primer array anterior sería un array de 4 por 3 y el siguiente sería un array de 2x2x2. Como puedes observar esta notación a partir de 3 dimensiones ya es muy liosa y normalmente no se usa. Utilizando esta notación también podemos inicializar rápidamente arrays irregulares, simplemente poniendo separados por comas los elementos que tiene cada dimensión:

```
int[][] i2d={{0,1},{0,1,2},{0,1,2,3},{0,1,2,3,4},{0,1,2,3,4,5}};
int[][][] i3d={ { {0,1},{0,2} } , {{0,1,3}} , {{0,3,4},{0,1,5} } };
```

Es posible que en algunos libros y en Internet se refieran a los arrays usando otra terminología. A los arrays unidimensionales es común llamarlos también **arreglos** o **vectores**, a los arrays de dos dimensiones es común llamarlos directamente **matrices**, y a los arrays de más de dos dimensiones es posible que los veas escritos como **matrices multidimensionales**. Sea como sea, lo más normal en la jerga informática es llamarlos arrays, término que procede del inglés.

**Autoevaluación**

**Dado el siguiente array:** `int[][][] i3d={{{0,1},{0,2}}, {{0,1,3}},{{0,3,4},{0,1,5}}};`

**¿Cuál es el valor de la posición [1][1][2]?**

- 3
- 4
- 5
- Ninguno de los anteriores.

**Para saber más**

Las matrices tienen asociadas un amplio abanico de operaciones (transposición, suma, producto, etc.). En la siguiente página tienes ejemplos de como realizar algunas de estas operaciones, usando por supuesto arrays:

Operaciones básicas con matrices.