Taller de preparación para la OCI - PUC

Arreglos Multidimensionales

1. Introducción

Cómo hemos aprendido hasta ahora, si queremos una colección de elementos debemos utilizar arreglos. Tal como almacenamos datos de tipo `int` o `String`, también podemos almacenar otros arreglos. A este tipo de arreglos se les llama arreglos multidimensionales.

Los **arreglos multidimensionales** nos abren nuevas formas de representar situaciónes. En especial, las que tengan forma de **tablero**. Por ejemplo, para representar un tablero de ajedrez se busca tener una estructura que te permita poner una ficha en una cuadrilla del tablero. Para esto se puede crear un arreglo bidimensional que al tablero.

Al igual de que los arreglos unidimensionales, los arreglos bidimensionales o matrices deben tener un largo fijo, es decir, uno debe decir cuántos arreglos internos tendrá. A su vez, todos los arreglos internos deben contener el mismo tipo de dato. Por ejemplo, una vez que declaramos que nuestra matriz es para almacenar `int`, **no** podemos agregarle ni `String`, `float`, `double` o cualquier otro tipo de dato que no sea `int`.

2. Estructura

2.1. Declarando matrices

Para usar un arreglo bidimensional, primero hay que declarar una variable que pueda referenciarlo. A modo general se declara de esta forma:

```
1  tipoDeDato[][] nombreDeVariable;
2
3  // Como por ejemplo, se declara una variable que
4  // es un arreglo de arreglos de doubles:
5
6  double[][] miLista;
```

A diferencia de un arreglo unidimensional, ahora debemos declarar la variable utilizando dos pares de paréntesis en vez de uno solo. Estos indican que nuestra variable eventualmente tendrá cierta cantidad de arreglos internos y cierta cantidad de elementos en cada arreglo interno.

2.2. Creando arreglos

El modo general de crear un arreglo y asignarlo a una variable **ya creada** es el siguiente:

```
nombreVariable = new tipoDeDato[cantidadArreglos][tamañoArreglos];

/* Si queremos crear un arreglo bidimensional
que contenga 3 arreglos de 5 doubles cada uno
y asignarlo a la variable ya declarada
se haría de la siguiente forma: */

double[][] miMatriz;
miMatriz = new double[3][5];

// acá estamos creando y declarando un arreglo bidimensional
double[][] miMatriz = new double[3][5];
```

Cuando se crea un arreglo, no es necesario darle el tamaño de los arreglos internos. Si se da un valor, todos los arreglos tendrán el mismo tamaño, en el caso contrario, se crearán variables de tipo `null` y será necesario entregarle los de forma separada.

2.3. Accediendo a los datos del arreglo bidimensional

Para acceder a los elementos de la matriz vamos a necesitar dos índices, uno que indique en cuál arreglo interno se encuentra el elemento y otro para indicar la posición dentro del arreglo interno. Los índices comenzarán desde el 0 y terminarán en uno menos que el tamaño del arreglo.

Siguiendo el ejemplo anterior, `miMatriz` tendrá 3 arreglos internos y 5 elementos en cada arreglo. Esto significa que podemos usar 3 índices para movernos entre arreglos internos, que irán desde el 0 hasta el 2. Asímismo, para movernos entre los elementos de algún arreglo interno, tendremos desde el índice 0 hasta el índice 4.

Entonces, modificar un elemento será de la siguiente forma:

```
// largo será la cantidad de arreglos
int largo = 3;
// ancho será la cantidad de elementos de un arreglo
int ancho = 4;
// creamos la matriz de "largo x ancho" elementos
int[][] miMatriz = int[largo][ancho];

// accedemos al primer arreglo
miMatriz[0];
// accedemos al primer elemento del primer arreglo
miMatriz[0][0];
// accedemos al último elemento del último arreglo
miMatriz[largo-1][ancho-1];
// accedemos al valor del segundo arreglo y tercer elemento
miMatriz[1][2];
```

2.4. Modificando los datos del arreglo

Para modificar los datos de un arreglo bidimensional, tenemos que acceder a la posición a cambiar y asignar un nuevo valor. También, existe la opción de cambiar un arreglo interno por completo[1]:

```
// Para cambiar el valor de un elemento
nombreVariableMatriz[índiceArreglo][índiceElemento] = nuevoValor;
// Si en el ejemplo anterior queremos que el
//dato que se encuentra en el tercer arreglo y segunda posición
//sea de valor 10:
miMatriz[2][1] = 10;

// Si queremos cambiar un arreglo interno
nombreVariableMatriz[índiceArreglo] = nuevoArreglo;
// Siguiendo el ejemplo anterior, si queremos que
// el primer arreglo solo contenga 1, entonces:

int[] nuevoArreglo = {1, 1, 1, 1};
miMatriz[0] = nuevoArreglo;
```

3. Operaciones

3.1. Iteración

Para iterar sobre un arreglo, necesitaremos dos `for` o `while`, uno que nos ayude a movernos por los arreglos internos y otro para iterar sobre los elementos. A continuación se entrega un código que imprime todos los elementos de un arreglo bidimensional:

```
double[][] miMatriz = {{1.9, 2.9}, {3.4, 3.5, 1.7}};

// el índice i nos va a servir para iterar
// sobre los arreglos internos
for (int i = 0; i < miMatriz.length; i++) {

  // el índice j será para iterar sobre
  // todos los elementos del arreglo
  // que se encuentra en la posición "i"

  for(int j = 0; j < miMatriz[i].length; j++){
    // imprimimos el valor
    System.out.print(miMatriz[i][j]+" ");
  }
  // imprimimos un salto de línea
  System.out.println();
}
// el resultado es:
// 1.9 2.9
// 3.4 3.5 1.7
```

Es muy importante entender que durante todas las iteraciones del `for` interno (el que usa el índice `j`) el índice `i` se mantiene fijo. Esto nos permite posicionarnos en un arreglo e iterar sobre los elementos de este. Una vez que termina el `for` interno, imprimimos un salto de línea para separar visualmente

[1] Siempre y cuando contenga elementos del mismo tipo que la matriz.

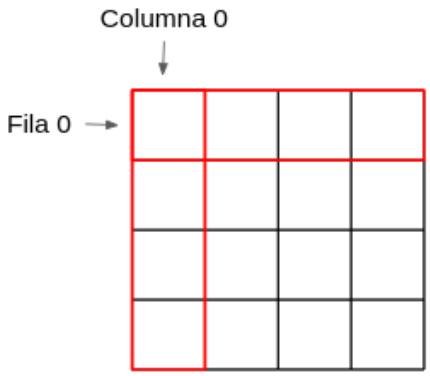
los elementos de un arreglo de otro, es por esto que el `2.9` y `3.4` están separados por un salto de línea. Luego aumenta el índice `i` en uno y se comienza a iterar sobre los elementos del siguiente arreglo, y sigue así hasta que termina de revisar todos los arreglos.

Ejemplo 3.1. Supongamos que para un trabajo del colegio te piden recolectar datos sobre la altura de personas de distintas edades. En específico, necesitas 5 datos para los rangos etarios de 20-25 años, 26-30 años, 31-35 años y de 36-40 años. Luego, te piden calcular el promedio[2] de altura para cada rango de edad.

```
1  double[][] datos = {{1.70, 1.80, 1.65, 1.58, 1.73},
2                      {1.67, 1.74, 1.70, 1.82, 1.52},
3                      {1.58, 1.67, 1.89, 1.78, 1.74},
4                      {1.87, 1.65, 1.68, 1.77, 1.72}};
5
6  // el índice i nos va a servir para iterar
7  // sobre los datos de cada rango etario
8  for (int i = 0; i < datos.length; i++) {
9
10   // el índice j va a servir para
11   // obtener el dato dado un rango etario
12   double suma = 0;
13
14     for(int j = 0; j < datos[i].length; j++){
15       suma += datos[i][j];
16     }
17     double promedio = suma/datos[i].length;
18     System.out.print("El promedio de altura para el grupo " + i);
19     System.out.println(" es de " + promedio + " metros");
20 }
```

3.2. Grillas

Muchas veces los arreglos bidimensionales nos van a servir para representar grillas. Las grillas tienen una cantidad definida de columnas y filas. Cada arreglo interno representaría una fila, mientras que la colección de elementos en una posición determinada en todas las filas puede representar una columna. Es decir, si queremos la primera columna, es lo mismo que guardar todos los elementos que se encuentran en la primera posición de los arreglos internos. La siguiente imagen remarca la primera columna y fila:



[2]El promedio se calcula como la suma de todos los datos dividido por la cantidad de datos.

En general, podemos si instanciamos el siguiente arreglo bidimensional:

```
1  // Supongamos que largo y ancho son
2  // dos números enteros que representan el largo y ancho
3  // de la grilla
4
5  int[][] matriz = new int[largo][ancho]
```

Cuando accedamos a `matriz[i][j]`, estamos pidiendo el elemento en la fila i y la columna j.

Ejemplo 3.2. Para estudiar matemáticas quisiste imprimir una tabla de multiplicar partiendo desde el 1 al 10. Como no te gustaron las que encontraste en internet, decidiste crear un tablero con tus conocimientos de Java y guardarlo para tener acceso más rápido.

```
1  int[][] tablaMultiplicacion = new int[11][11]
2
3  for (int fila = 0; fila < tablaMultiplicacion.length; fila++) {
4
5      for(int col = 0; col < tablaMultiplicacion[fila].length; col++){
6        tablaMultiplicacion[fila][col] = fila*col;
7      }
8
9  }
```