Arrays101

Introducción

¿Qué es un array?

Un array es una colección de ítem, almacenados en posiciones de memoria contiguas. Dado que están almacenados juntos, acceder a la colección entera es sencillo y directo.

Crear un array

Los arrays pueden almacener N items, donde N es el valor que le queramos dar. A su vez, se debe especificar el tipo de dato que vamos a almacenar en el array.

Ej:

```
//Array de 15 números
int[] numeros = new int[15];
```

Acceder a los elementos

Cada una de las posiciones de los elementos se identifica usando un número entre 0 y N-1, donde la primera posición es el número 0, la segunda el 1, etc. Cada una de esas posiciones se llama índice(index).

Escribir elementos

Ej:

```
/**Imaginemos que queremos escribir un nuevo comic de Batman en nuestro array de comics,
que contiene el título, la fecha y el número de págs. **/

//Creamos el comic
Comic comicBatman = new Comic("Batman", 2002, 136);

comics[10] = comicBatman;

//En caso de que queramos sobrescribir una posición basta con pisar el valor anterior
Comic nuevoComicBatman = new Comic("Batman2", 2022, 150);
comics[10] = nuevoComicBatman;
```

Leer elementos

Ej:

```
//Podemos comprobar cualquier valor del array
System.out.println(comics[0]);
System.out.println(comics[5]);
System.out.println(comics[10]);
```

Escribir y leer elementos con un loop

Ej:

```
/** Imaginemos que queremos calcular el valor del cubo de los valores 0 a 9 */
int[] cubos = new int[10];

for(int i=0;i<10;i++){
  cubos[i] = i*i*i;
}

//Con for
for(int i=0;i<10;i++){
  System.out.println(cubos[i]);
}

//Con foreach (no se necesita el index / facil lectura)
for(int cubo: cubos){
  System.out.println(cubo);
}
```

Length

Cuando creamos un array, lo inicializamos con un valor:

```
Comic[] comics = new Comic[10];
```

Este valor inicial, que no puede ser cambiado posteriormente, es el valor máximo de objetos que va a poder almacenar(Comics, en este caso).

Se puede comprobar este valor con la propiedad length:

```
System.out.println(comics.length);
```

En caso de que queramos llevar una cuenta de cuantos objetos hay actualmente en el array, debemos llevar la cuenta nosotros mismo, con un contador por ejemplo:

```
int[] numeros = new int[10];
int contador = 0;

//Añadimos 3 numeros al array
for(int i=0;i<3;i++){
  numeros[i] = i*2;
  contador++;
}

//Comprobaciones
System.out.println("Tamaño: " + numeros.length);
System.out.println("Introducidos: " + contador);
```

Normalmente recibiremos como parámetro de una función un array, del cual no sabemos su tamaño, por lo que podemos asumir que el array es del tamaño justo y todos los índices contienen un valor, y podemos trabajar con él de la siguiente manera:

```
/** Imaginemos que queremos devolver el valor mayor de un array.
Tenemos una funcion que recibe el array de numeros**/

public int miFuncion(int[] numeros){
  int mayor = 0;
  //Importante el i<numeros.length dado que los indices empiezan en 0
  for(int i=0;i<numeros.length;i++){
    if(numeros[i] > mayor){
      mayor = numeros[i];
    }
  }
  return mayor;
}
```

Ejercicios

maxConsecutiveOnes.java

findNumbersWithEvenNumberofDigits.java

squaresofSortedArray.java

Insertar elementos

Insertar al final

Dado que tenemos nuestra variable con la cantidad de valores que hemos insertado, lo único que debemos hacer es asignar un valor en esa posición.

Ej:

```
int[] nums = new int[6];
int insertados = 0;

//En forma de bucle
for(int i=0;i<3;i++){
  nums[insertados] = i;
  insertados++;
}
```

```
//Asignacion simple
nums[insertados] = 10;
```

Insertar al inicio

Para insertar un elemento al principio del array, lo que debemos hacer es mover todos los demás elementos una posición a la derecha (su posición original +1).

Debemos empezar por el final para evitar sobrescribir los elementos.

Ej:

```
//Imaginemos que tenemos 3 elementos en nuestro array y queremos añadir un cuarto al inicio
for(int i=3;i>=0;i--){
  nums[i+1] = nums[i];
}
nums[0] = 20;
```

Insertar en cualquier posición

Como en el caso anterior, debemos mover los elementos una posición a la derecha hasta llegar a la posición que queramos, una vez creado el espacio, insertamos el valor.

Ej:

```
//Imaginemos que queremos introducir un elemento en la posición 2 de un array con 4 elementos
for(int i=4;i>=2;i--){
  nums[i+1] = nums[i];
}
//Ya tenemos el espacio creado, por lo que podemos insertar el valor ya
nums[2] = 20;
```

Ejercicios

duplicateZeros.java

mergeSortedArray.java

Borrar elementos

Borrar al final

Borrar al final del array es el caso que menos tiempo consume, dado que borrar el último elemento no repercute sobre los demás.

Ej:

```
int[] nums = new int[];
int insertados = 0;

//Añadimos 6 numeros a nuestro array
for(int i=0;i<6;i++){
  nums[i] = i+3;
  insertados++;
}

//Para "borrar" un elemento es tan simple como:
insertados--;

//Este borrado no es un borrado de por si ya que si hacemos:
for(int i=0;i<nums.length;i++){
  System.out.println(nums[i]);
}
//El elemento que acabamos de eliminar sigue existiendo, por lo que debemos hacer es
//iterar por nuestra variable insertados:
for(int i=0;i<insertados;i++){
  System.out.println(nums[i]);
}
```

Borrar al principio

Para borrar el primer elemento de un array, lo que debemos hacer es desplazar todos los demás elementos a la izquierda.

Ej:

```
//Imaginemos que tenemos nuestro array nums lleno
//Empezamos en 1 dado que la posición 0 es la que queremos borrar
for(int i=1;i<nums.length;i++){
  nums[i-1] = nums[i];
}

//Importante reducir el valor de la variable para mantener la consistencia
insertados--;
```

Borrar en cualquier posición

Para borrar en cualquier posición, lo que debemos hacer es rellenar el hueco que ha dejado, desplazando a la izquierda todos los elementos a continuación del elemento borrado.

Ej:

```
//Imaginemos que tenemos nuestro array nums lleno y queremos borrar
//el número que se encuentra en la posición 4
//Empezamos en 5 ya que es la posición siguiente a la que queremos borrar
for(int i=5;i<nums.length;i++){
  nums[i-1] = nums[i];
}

insertados--;
```

Ejercicios

removeElement.java

removeDuplicatesSortedArray.java

Buscar elementos en un array

Búsqueda lineal

En caso de que el índice sea desconocido, como es en la mayoría de casos, se comprueba cada elemento hasta que encontremos el que estamos buscando, o lleguemos al final del array

Ej:

```
//Imaginemos que queremos buscar un elemento element, pero tendremos en cuenta también
//los casos extremos, como que el array sea nuelo o no encontremos el elemento

public static boolean busquedaLineal(int[] array, int tam, int element){
  if(array == null || length == 0){
    return false;
  }

  for(int i=0;i<tam;i++){
    if(array[i] == element){
      return true;
    }
```

```
  }

  return false;
}
```

Ejercicios

checkNandDouble.java

validMountainArray.java

Operaciones in-place

Introducción

Comúnmente, se espera minimizar el tiempo y el espacio de complejidad de la implementación, a lo cual ayudan las operaciones in-place.

Ej:

> Dado un array de integers, devuelve un array donde cada elemento en un índice para sea elevado al cuadrado.

```
Input: array = [9, -2, -9, 11, 56, -12, -3]
Output: [81, -2, 81, 11, 3136, -12, 9]
```

> Dos soluciones: la primera crear un array del primer tamaño que el original, donde copiaremos los elementos en una posición impar, y elevaremos los elementos en una posición par.

```
public int[] squareEven(int[] array, int length) {

  // Check for edge cases.
  if (array == null) {
    return null;
  }

  // Create a resultant Array which would hold the result.
  int result[] = new int[length];

  // Iterate through the original Array.
  for(int i = 0; i < length; i++) {

    // Get the element from slot i of the input Array.
```

```
    int element = array[i];

    // If the index is an even number, then we need to square element.
    if (i % 2 == 0) {
      element *= element;
    }

    // Write element into the result Array.
    result[i] = element;
  }

  // Return the result Array.
  return result;
}
```

Sin embargo, esta es una forma ineficiente, ya que usa O(length) de espacio extra.

Lo correcto sería iterar sobre el array original, sobrescribiendo los índices con valor par. Está técnica de trabajar sobre el propio array es lo que se denomina operaciones in-place.

```
public int[] squareEven(int[] array, int length) {

  // Check for edge cases.
  if (array == null) {
    return array;
  }

  // Iterate through the original array.
  for(int i = 0; i < length; i++) {

    // If the index is an even number, then we need to square the element
    // and replace the original value in the Array.
    if (i % 2 == 0) {
      array[i] *= array[i];
    }
    // Notice how we don't need to do *anything* for the odd indexes? :-)
  }

  // We just return the original array. Some problems on leetcode require you
  // to return it, and other's dont.
  return array;
}
```

Ejercicio

replaceElementsGreatestElementRight.java