ARRAYS

MÓDULO: Programación en Java

ÍNDICE DE CONTENIDOS

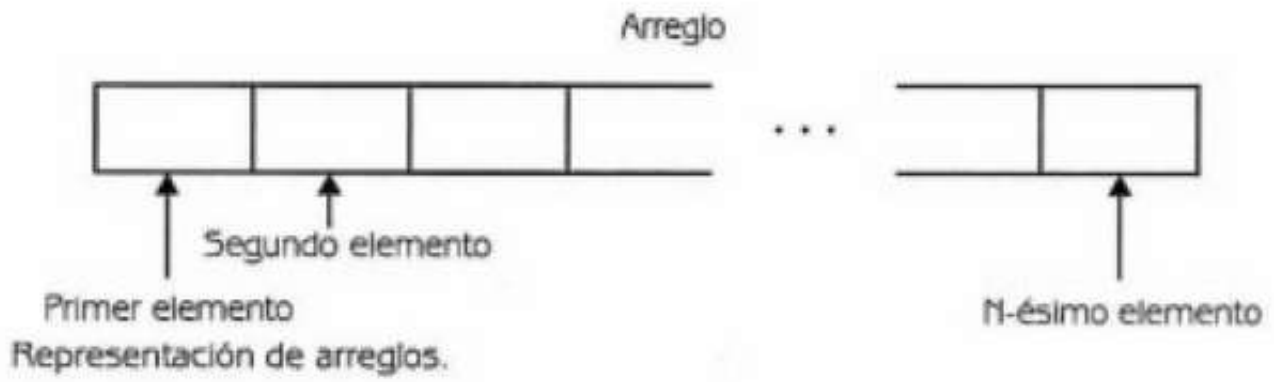
1.OBJETIVOS

- Conocer la estructura de tipo array para almacenar una lista de valores.
- Conocer sus funcionalidades más interesantes.
- Aprender a crear Arrays de diferentes tipos.
- Analizar el flujo de código y ser capaz de decidir el uso de esta estructura.

2. ARRAYS

Un arreglo se define como una colección finita, homogénea y ordenada de elementos.

- **Finita**: todo arreglo tiene un límite, es decirse debe determinar cuál será el número máximo de elementos que podrán formar parte del arreglo
- **Homogénea**: todos los elementos de un arreglo son del mismo tipo (todos enteros, todos reales, etc., pero nunca una combinación de distintos tipos).
- **Ordenada**: se puede determinar cuál es el primer elemento, el segundo, el tercero,... y el n-ésimo elemento.



Representación de arreglos.

2. ARRAYS

Un arreglo tiene la característica de que puede almacenar a N elementos del mismo tipo y además permite el acceso a cada uno de estos elementos. Así, se distinguen los siguientes componentes:

- Los componentes.

- Los índices. Los componentes hacen referencia a los elementos que componen o forman el arreglo. Es decir, son los valores que se almacenan en cada una de sus casillas. Los índices, por otra parte, son los que permiten accesar a los componentes del arreglo en forma individual. Para hacer referencia a un componente de un arreglo se necesita:

- El nombre del arreglo.

- El índice del elemento.

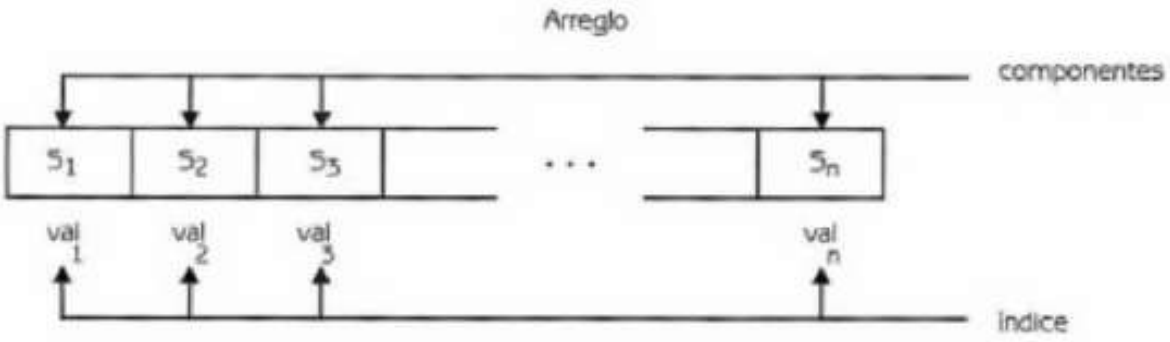


Figura 4.2 Indices y componentes de un arreglo.

3.DECLARACIÓN DE ARRAYS

Como no es nuestra intención seguir la sintaxis de algún lenguaje de programación en particular, definimos un arreglo de la siguiente manera:

ident_arreglo = ARREGLO [líminf .. límsup] DE tipo

Con los valores líminf y límsup se declara el tipo de los índices así como el número de elementos que tendrá el arreglo. El número total de elementos (NTE) que tendrá el arreglo puede calcularse con la fórmula 4.1:

$$NTE = límsup - líminf + 1$$

3.DECLARACIÓN DE ARRAYS

Con tipo se declara el tipo de datos para todos los elementos del arreglo. El tipo de los elementos no tiene que ser necesariamente el mismo que el de los índices.

Observaciones

- El tipo del índice puede ser cualquier tipo ordinal (caracter, entero, etc.).
- El tipo de los componentes puede ser cualquier tipo (entero, real, cadena de caracteres, registro, arreglo, etc.).
- Se utilizan los corchetes "[]" para indicar el índice de un arreglo. Entre los [] se debe escribir un valor ordinal (puede ser una variable, una constante o una expresión tan compleja como se quiera, pero que dé como resultado un valor ordinal). Se verán a continuación algunos ejemplos de arreglos:

3.DECLARACIÓN DE ARRAYS

Sea ARRE un arreglo de 70 elementos enteros con índices enteros. Su representación queda como se muestra en la figura 4.3.

ARRE = ARREGLO [1. .70] DE enteros

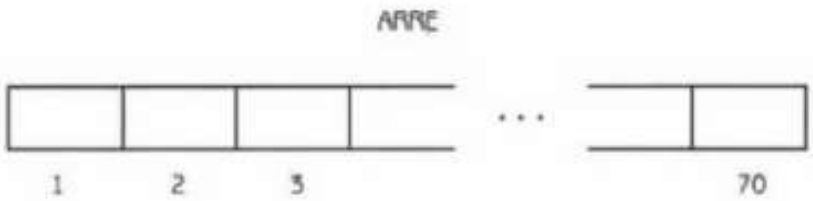


Figura 4.3

- NTE = (70 – 1 + 1) = 70
- Cada elemento del arreglo ARRE será un número entero y podrá accesarse por medio de un índice que será un valor comprendido entre 1 y 70.

Así por ejemplo:

ARRE[1] hace referencia al elemento de la posición 1.
ARRE[2] hace referencia al elemento de la posición 2.
. . .
. . .
ARRE[70]hace referencia al elemento de la posición 70.

4.OPERACIONES CON ARRAYS

A continuación se presentan las operaciones más comunes en arreglos:

- Lectura/Escritura.

- Asignación.

- Actualización:
  - Inserción.
  - Eliminación.
  - Modificación.

- Ordenación.

- Búsqueda. Como los arreglos son datos estructurados, muchas de estas operaciones no pueden llevarse a cabo de manera global, sino que se debe trabajar sobre cada elemento. A continuación se analizará cada una de estas operaciones. Ordenación y búsqueda, serán analizadas en los problemas que presentaremos posteriormente.

4.OPERACIONES CON ARRAYS

4.1. LECTURA

Lectura. El proceso de lectura de un arreglo consiste en leer y asignar un valor a cada uno de sus elementos.

Leer ARRE[1],

Leer ARRE[2],

. . .

Leer ARRE[70]

ARRE[1], ARRE[2], . . . ,ARRE[70]

De esta forma no resulta práctico, por lo tanto se usará un ciclo para leer todos los elementos del arreglo.

```
.
.
Hacer I ←— 1
Repetir con I desde 1 hasta 70
        Leer ARRE[I]
        Hacer I ←— I + 1
{Fin del ciclo}
.
.
```

4.OPERACIONES CON ARRAYS

4.2. ASIGNACIÓN

En general no es posible **asignar** directamente un valor a todo el arreglo, sino que se debe asignar el valor deseado a cada componente.

En seguida se analizan algunos ejemplos de asignación. En los dos primeros casos se asigna un valor a una determinada casilla del arreglo (en el primero a la señalada por el índice 1 y en el segundo a la indicada por el índice 3).

ARRE [1] ⟵ 120
ARRE [3] ⟵ ARRE [1]/4

4.OPERACIONES CON ARRAYS

4.3. ACTUALIZACIÓN

Para actualizar un elemento en un array, simplemente se asigna un nuevo valor al índice correspondiente del array.

```
public class ActualizarArray {
   public static void main(String[] args) {
      int[] numeros = {5, 3, 8, 1, 9, 2, 7, 4, 6, 0};
      // Mostrar array original
         System.out.println("Array original: ");
          printArray(numeros);
      // Actualizar el valor del tercer elemento (índice 2)
           numeros[2] = 10;
      // Mostrar array actualizado
           System.out.println("Array actualizado: ");
         printArray(numeros);
       }

   public static void printArray(int[] arr) {
      for (int num : arr) {
         System.out.print(num + " ");
      }
      System.out.println();   }}
```

4.OPERACIONES CON ARRAYS

4.3. ACTUALIZACIÓN

Podemos utilizar un bucle para actualizar todos los elementos de un array. Por ejemplo, podemos incrementar cada elemento en 1.

```
public class IncrementarArray {
    public static void main(String[] args) {
        int[] numeros = {5, 3, 8, 1, 9, 2, 7, 4, 6, 0};

        // Mostrar array original
        System.out.println("Array original: ");
        printArray(numeros);

        // Incrementar cada elemento en 1
        for (int i = 0; i < numeros.length; i++) {
            numeros[i] += 1;
        }

        // Mostrar array actualizado
        System.out.println("Array incrementado: ");
        printArray(numeros);
    }

    public static void printArray(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();   }}
```

4.OPERACIONES CON ARRAYS

4.3. ACTUALIZACIÓN

El método Arrays.fill() se puede usar para establecer todos los elementos de un array a un valor específico.

```
import java.util.Arrays;

public class RellenarArray {
    public static void main(String[] args) {
        int[] numeros = new int[10];

        // Rellenar el array con el valor 7
        Arrays.fill(numeros, 7);

        // Mostrar array rellenado
        System.out.println("Array rellenado: " + Arrays.toString(numeros));
    }
}
```

4.OPERACIONES CON ARRAYS

4.3. ACTUALIZACIÓN

Si se necesita cambiar el tamaño del array, se puede crear un nuevo array y copiar los elementos del array original al nuevo array, luego actualizar el nuevo array según sea necesario.

```
import java.util.Arrays;

public class CopiarYActualizarArray {
    public static void main(String[] args) {
        int[] numeros = {5, 3, 8, 1, 9, 2, 7, 4, 6, 0};

        // Crear un nuevo array con tamaño mayor
        int[] nuevoArray = Arrays.copyOf(numeros, numeros.length + 5);

        // Actualizar los nuevos elementos
        for (int i = numeros.length; i < nuevoArray.length; i++) {
            nuevoArray[i] = i * 2;  // Ejemplo: rellenar con valores
        }

        // Mostrar el nuevo array
        System.out.println("Nuevo array actualizado: " + Arrays.toString(nuevoArray));
    }
}
```

4.OPERACIONES CON ARRAYS

4.4. ORDENACIÓN

Ordenar arrays en Java es una tarea común que puede realizarse de varias formas, utilizando tanto métodos de la biblioteca estándar como algoritmos de ordenación personalizados. A continuación, se presentan algunos métodos para ordenar arrays en Java:

Java proporciona un método de ordenación rápido y eficiente llamado Arrays.sort(). Este método utiliza un algoritmo híbrido de Quicksort, Timsort y Merge Sort.

Ej:

```
import java.util.Arrays;

public class OrdenarArrays {
    public static void main(String[] args) {
        int[] numeros = {5, 3, 8, 1, 9, 2, 7, 4, 6, 0};

        // Mostrar array original
        System.out.println("Array original: " + Arrays.toString(numeros));

        // Ordenar el array
        Arrays.sort(numeros);}}
```

4.OPERACIONES CON ARRAYS

4.5. BÚSQUEDA

La búsqueda de valores en un array en Java puede realizarse de varias maneras, dependiendo de los requisitos específicos del problema, como si el array está ordenado o no. A continuación, se presentan ejemplos de diferentes métodos para buscar valores en un array:

La búsqueda binaria es un método más eficiente que requiere que el array esté ordenado. Este método divide el array repetidamente en mitades hasta encontrar el valor buscado.

```
public class BusquedaBinaria {
   public static void main(String[] args) {
      int[] numeros = {5, 3, 8, 1, 9, 2, 7, 4, 6, 0};
      int valorBuscado = 7;

      // Ordenar el array antes de realizar la búsqueda binaria
      Arrays.sort(numeros);
      int indice = Arrays.binarySearch(numeros, valorBuscado);
      if (indice >= 0) {
         System.out.println("El valor " + valorBuscado + " se encuentra en el índice: " + indice);
      } else {
         System.out.println("El valor " + valorBuscado + " no se encuentra en el array.");
      }   }}
```

RESUMEN DE LA UNIDAD

- Un array en Java es una estructura de datos que permite almacenar múltiples elementos del mismo tipo, y se declara especificando el tipo de los elementos seguido de corchetes.
- Los elementos de un array se acceden mediante índices numéricos, comenzando desde cero, lo que permite obtener o modificar el valor de cualquier posición específica .
- El tamaño de un array en Java es fijo y no puede cambiar, lo que significa que no se pueden añadir o eliminar elementos después de su creación..
- Java soporta arrays multidimensionales, permitiendo la creación de matrices y otros arreglos más complejos, donde cada elemento puede ser a su vez otro array.

BIBLIOGRAFÍA/WEBGRAFÍA

- Thierry Richard. “Los fundamentos del lenguaje Java”.
- Vozmediano,A.M. "Java para novatos”.
- Jimenez, Alfonso.”Aprender a programar en Java”.
- Gervás, Luc. “Aprender los fundamentos del lenguaje Java”

RECURSOS DE INTERÉS

- **Arrays en Java:** https://encr.pw/870nu
- **Funciones y métodos del objeto Array en Java:** https://acesse.dev/xZSqp
- **¿Qué es un Array en Java?:** https://acesse.dev/NmNQK
- **Ordenar Arrays en Java:** https://l1nq.com/tEUX8