5 | Arreglos

Manuel Alcántara Juárez
Verónica Esther Arriola Ríos

Meta

Que el alumno aprenda a plantear problemas donde se utilicen arreglos como estructura para el manejo de los datos.

Objetivos

Al finalizar la práctica el alumno será capaz de:

1. Crear arreglos de tipos primitivos.
2. Agregar, consultar y manipular elementos de un arreglo de forma individual o dentro de ciclos.
3. Utilizar arreglos para resolver problemas donde esta estructura sea adecuada.

Código Auxiliar 5.1: Arreglos

https://github.com/computacion-ciencias/icc-arreglos

Antecedentes

Ciclos

Además de las instrucciones de selección (`if` y `switch`) y repetición (`while`, `for`, `do... while`), Java proporciona las instrucciones `break` y `continue` para alterar el flujo de control.

Instrucción `break`

Esta instrucción, al ejecutarse en un ciclo `while`, `for`, `do...while` o `switch`, ocasiona la salida inmediata de esa instrucción. La ejecución continúa con la primera instrucción que va después de la instrucción de control.

En el siguiente ejemplo queremos que el programa cuente los dígitos de un número pero solamente hasta llegar a 5 dígitos, porque después se cansa, en caso de que tenga más dejaremos de contarlos:

Listado 5.1: CuentaDígitos.java

```
1  public class CuentaDígitos {
2    public static void main(String args[]){
3      int número = Integer.parseInt(args[0]);
4      int dígitos = 0;
5      while(número > 0) {
6        número /= 10;
7        dígitos++;
8        if (dígitos == 5) {
9          break;            // Interrumpe la ejecución del ciclo
10       }
11       if (dígitos == 5) {
12         // Este bloque nunca se ejecuta
13         System.out.println("El número tiene 5 o más dígitos.");
14       }
15     }
16     System.out.println("Conté hasta " + dígitos + " dígitos");
17   }
18 }
```

```
$ javac CuentaDígitos.java
$ java CuentaDígitos 4557888
Conté hasta 5 dígitos.
$ java CuentaDígitos 455
Conté hasta 3 dígitos.
```

Este programa cuenta los dígitos de un número y en caso de que dicho número tenga 5 o más dígitos intenta mostrar por consola el mensaje "El número tiene 5 o más dígitos", sin embargo el mensaje nunca se muestra. Véase la línea `if (dígitos == 5) break;` cuya presencia hace que en caso de que el número haya llegado a los 5 dígitos se rompa la iteración del ciclo y se continúe con las instrucciones que van después del bloque `while`.

Instrucción `continue`

Esta instrucción, al ejecutarse en un ciclo `while`, `for`, `do...while` o `switch`, evita las instrucciones restantes en el cuerpo del ciclo y procede con la siguiente iteración del

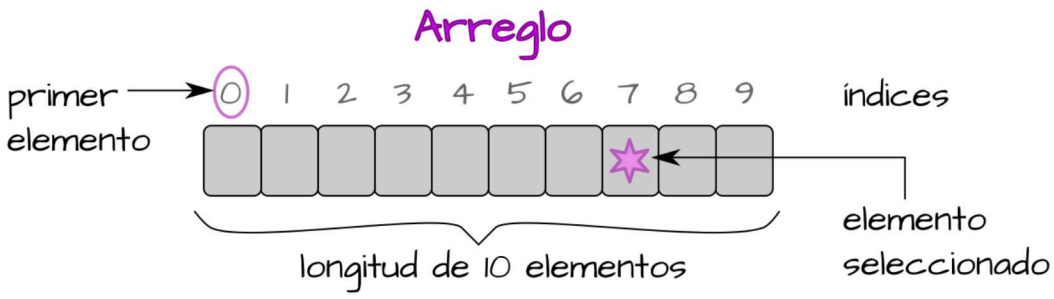

Figura 5.1 Arreglo.

mismo. En las instrucciones `while` y `do...while`, el programa evalúa la condición de continuación de ciclo inmediatamente después de que se ejecuta la instrucción `continue`. En una instrucción `for` se ejecuta la expresión de incremento y luego el programa evalúa la condición de continuación del ciclo. Por ejemplo, en el fragmento de código siguiente:

```
for(int j = 0; j < 10; j++){
  sentencia 1;
  sentencia 2;
  sentencia 3;
  continue;
  sentencia 4;
}
```

el ciclo se ejecuta 10 veces, pero con la salvedad de que la sentencia 4 no se ejecuta ninguna vez. Es decir, se ejecutan las sentencias 1, 2 y 3 y cuando se llega a la sentencia de control `continue`, se incrementa `j` y se vuelve a comprobar la condición del `for`, en caso de cumplirse de nuevo se ejecuta la sentencia 1 y así sucesivamente.

Arreglos

Los arreglos son estructuras de datos (colecciones de elementos de datos relacionados) que contienen datos del mismo tipo. Los arreglos son entidades de longitud fija; conservan siempre la misma longitud una vez creados, aunque la referencia a un arreglo puede reasignarse a un nuevo arreglo de distinta longitud.

En Java, un arreglo es un grupo de variables (llamadas elementos o componentes) que contienen valores del mismo tipo Figura 5.1. Los arreglos en Java son objetos, por lo que se consideran como tipo de referencia. Los elementos de un arreglo pueden ser de tipo primitivo o de referencia.

Declaración y creación de arreglos

Los objetos arreglo ocupan espacio en memoria. Todos los objetos en Java (incluyendo los arreglos) deben crearse con la palabra new. Para crear un arreglo, el programador especifica el tipo de cada elemento y el número de elementos que se requieren para el arreglo siguiendo la siguiente sintaxis:

```
<creación_arreglo> ::= <tipo> <identificador>[] = new <tipo>[<int>];
```

El entero que se coloca entre corchetes es el número de elementos.

Por ejemplo, la siguiente instrucción crea un arreglo de enteros llamado c:

```
1 int c[] = new int[12];
```

Al crear un arreglo, cada uno de sus elementos recibe un valor predeterminado: cero para los elementos numéricos de tipos primitivos, false para los elementos boolean y null para las referencias.

Un programa puede acceder a un elemento del arreglo mediante un índice. En los arreglos los índices van de 0 a la longitud del arreglo menos 1 y éstos deben ser expresiones enteras positivas que van en los corchetes.

```
1 int x = c[5];    // Lee el valor en la posición 5, lo guarda en x
2 c[6] = 10;       // Asigna el valor 10 en la posición 6
```

Como ya habíamos dicho anteriormente, los arreglos tienen longitud fija y podemos saberla mediante la variable length, que le pertenece al arreglo. Podemos conocer su valor mediante el operador punto (.) de la siguiente manera:

```
1 c.length
```

Lo que nos devolverá un valor entero positivo, en este caso, 12.

Inicialización de arreglos

En Java es posible crear un arreglo e inicializar sus elementos en una sola instrucción mediante llaves de la siguiente manera:

```
1 Tipo nombreArreglo[] = {elemento_1, elemento_2, ... , elemento_n};
```

La longitud del arreglo se determina en base al número de elementos en la lista inicializadora. Esta declaración no requiere de new para crear el objeto arreglo. Por ejemplo, la declaración:

```
int n[] = {10, 20, 30};
```

crea un arreglo de cinco números enteros.

También es posible inicializar el arreglo de la siguiente manera:

```
int[] n = new int[]{10,20,30};
```

O:

```
int[] n = new int[3];
n[0] = 10;
n[1] = 20;
n[2] = 30;
```

Suma de los elementos de un arreglo

A menudo, los elementos de un arreglo representan una serie de valores que se emplearán en un cálculo. Un cálculo muy común es sumarlos de la siguiente manera:

Listado 5.2: DemoSuma.java

```
public class DemoSuma {

  /** Suma los elementos en <code>arreglo</code>. */
  public static int suma(int[] arreglo) {
    int total = 0;
    // sumar el valor de cada elemento al total
    for(int i = 0; i < arreglo.length; i++){
        total += arreglo[i];
    }
    return total;
  }

  public static void main(String[] args) {
    int arreglo[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int total = suma(arreglo);
    System.out.println("La suma es: " + total);
  }
}
```

Búsqueda en arreglos

A menudo los programadores trabajan con grandes cantidades de datos almacenadas en arreglos, por lo que a veces es necesario determinar si un arreglo contiene un valor

que coincide con cierto valor clave. A este proceso le llamamos búsqueda y la más básica es la búsqueda lineal.

El algoritmo de búsqueda lineal usa un ciclo `for` que contiene una instrucción `if` para iterar sobre los elementos del arreglo y comparar cada elemento con una clave de búsqueda, es decir, el valor a localizar en el arreglo. Si se encuentra esta clave, se devuelve el índice del elemento, el cual nos dirá su ubicación exacta en el arreglo. La implementación del algoritmo en Java es la siguiente:

Listado 5.3: DemoBusca.java

```
1  public class DemoBusca {
2
3    /** Busca el valor <code>clave</code> en <code>arr</code>. */
4    public static int busca(int[] arr, int clave) {
5      // iterar a través de los elementos del arreglo
6      for(int contador = 0; contador < arr.length; contador++){
7        // si el elemento del arreglo es igual al valor de la clave,
8        // devuelve su índice
9        if(arr[contador] == clave) {
10         // Se interrumpe la función regresando el valor
11         return clave;
12       }
13     }
14     return -1;          // No encontró la clave
15   }
16
17   public static void main(String[] args) {
18     int arreglo[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
19     int posición = busca(arreglo, 7);
20     System.out.println("Se encontró en la posición: " + posición);
21   }
22 }
```

ArrayIndexOutOfBoundsException

Ojo, cuando utilizes arreglos y trates de acceder a los elementos guardados en ellos, debes asegurarte de solicitar un índice válido. Si solicitas un índice negativo o una posición mayor o igual al tamaño del arreglo, Java te lanzará una excepción tipo:

`ArrayIndexOutOfBoundsException`.

Desarrollo

Para esta práctica deberás escribir dos programas realizando los ejercicios siguientes:

1. Se quiere saber la frecuencia con que caen los números aleatorios en Java. Para ello:

   a) Crea un programa que simule tiradas de un hiperdado de 50 caras del 1 al 50. Para generar números aleatorios en Java puedes utilizar los comandos siguientes:

```java
import java.util.Random;

public class DemoRandom {

  public static void main(String[] args) {
    // Crea un objeto que genera números aleatorios
    Random generador = new Random();

    // Genera enteros en [0, 10).
    int rnd1 = generador.nextInt(10);
    int rnd2 = generador.nextInt(10);

    System.out.println("Se generaron: " + rn1 + " y " +
        ↪ rnd2);
  }
}
```

   b) Simula 10,000 lanzamientos. Utiliza un arreglo del tamaño y tipo correcto para llevar la cuenta de cuántas veces ha aparecido cada número[1].

   c) Imprime en pantalla la frecuencia de cada cara.

   d) Verifica que el conteo fue correcto haciendo la suma de las frecuencias almacenadas en el arreglo. La suma debe dar 10,000.

   La salida de tu programa debe de parecerse a esto:

```
$ javac icc.arreglos.Frecuencias.java
$ java icc.arreglos.Frecuencias
Cara Salida
1    125
2    144
3    116
...
50   119
La suma de las frecuencias es = 10,000
```

2. La *media* y la *variancia* nos ayudan a analizar un conjunto de valores y se utilizan mucho en Estadística. La media μ es el promedio de los datos y se calcula de la

[1] A esto se le llama frecuencia.

siguiente manera:

$$\mu = \frac{1}{N} \sum_{i=0}^{N-1} a_i \tag{5.1}$$

donde N es el número de elementos y a_i es el i-ésimo dato de entrada.

La varianza σ^2 caracteriza la dispersión de los datos[2] y requiere a la media para su cálculo:

$$\sigma^2 = \frac{1}{N} \sum_{i=0}^{N-1} (a_i - \mu)^2 \tag{5.2}$$

a) Crea un programa que pida un entero n al usuario, el cuál será el número de elementos a introducir. Luego ve pidiendo esos n elementos para llenar un arreglo de ese tamaño.

b) Crea una función que dado un arreglo nos calcule y regrese la media de sus datos.

c) Crea una función que dado un arreglo y la media nos calcule la varianza.

d) Invoca a estas dos funciones para calcular la media y varianza del conjunto de valores proporcionados e imprime los resultados.

La salida debe parecerse a esto:

Listado 5.4: Programa encontrar media y varianza

```
$ javac icc.arreglos.Estadísticas.java
$ java icc.arreglos.Estadísticas
Proporciona el número de datos a proporcionar: 4
Dame el 1 valor: 2
Dame el 2 valor: 6
Dame el 3 valor: 0
Dame el 4 valor: 8
La media es: 4
La varianza es: 10
```

Para pedirle los datos al usuario puedes usar un objeto de tipo `Scanner` dentro del método `main`.

Listado 5.5: demos/DemoScanner.java

```
1 package demos;
2
3 import java.util.Scanner;
4
5 public class DemoScanner {
```

[2]Es decir, qué tanto se alejan de la media.

```
6     public static void main(String[] args) {
7       Scanner sc = new Scanner(System.in);
8
9       System.out.println("Escribe un número entero:");
10      int i = sc.nextInt();
11
12      System.out.println("Escribe un texto:");
13      String texto = sc.next();
14
15      System.out.println("Lo que escribiste fue:");
16      System.out.println(i);
17      System.out.println(texto);
18    }
19  }
```

Entregables

En tu repo deberás añadir los archivos con tus dos programas. Por buenas prácticas de programación ambos deben estar dentro de un paquete, puedes elegir el nombre. Puedes usar o no `ant`, si lo usas incluye tu archivo `build.xml`; la otra opción es compilar y ejecutar tus ejemplos usando los comandos `javac` y `java`. En ambos casos explica cómo correr tus programas en el README.