Vectores y matrices

Arrays

- Declaración
- Creación
- Acceso a los elementos de un array
- Manipulación de vectores y matrices

Algoritmos de ordenación

- Ordenación por selección
- Ordenación por inserción
- Ordenación por intercambio directo (método de la burbuja)
- Ordenación rápida (QuickSort)

Algoritmos de búsqueda

- Búsqueda lineal
- Búsqueda binaria

Apéndice: Cadenas de caracteres

Arrays

Un array es una estructura de datos
que contiene una colección de datos del mismo tipo

Ejemplos

Temperaturas mínimas de los últimos treinta días

Valor de las acciones de una empresa durante la última semana

…

Propiedades de los arrays

- Los arrays se utilizan como **contenedores** para almacenar datos relacionados (en vez de declarar variables por separado para cada uno de los elementos del array).

- Todos los **datos** incluidos en el array son **del mismo tipo**. Se pueden crear arrays de enteros de tipo `int` o de reales de tipo `float`, pero en un mismo array no se pueden mezclar datos de tipo `int` y datos de tipo `float`.

- El tamaño del array se establece cuando se crea el array (con el operador `new`, igual que cualquier otro objeto).

- A los elementos del array se accederá a través de la posición que ocupan dentro del conjunto de elementos del array.

Terminología

Los arrays unidimensionales se conocen con el nombre de **vectores**.

Los arrays bidimensionales se conocen con el nombre de **matrices**.

Declaración

Para declarar un array,
se utilizan corchetes para indicar que se trata de un array
y no de una simple variable del tipo especificado.

Vector (array unidimensional):

```
tipo identificador[];
```

o bien

```
tipo[] identificador;
```

donde

`tipo` es el tipo de dato de los elementos del vector

`identificador` es el identificador de la variable.

Matriz (array bidimensional):

```
tipo identificador[][];
```

o bien

```
tipo[][] identificador;
```

NOTA: No es una buena idea que el identificador del array termine en un dígito, p.ej. vector3

Creación

Los arrays se crean con el operador `new`.

Vector (array unidimensional):

```
vector = new tipo[elementos];
```

Entre corchetes se indica el tamaño del vector.

`tipo` debe coincidir con el tipo con el que se haya declarado el vector.

`vector` debe ser una variable declarada como `tipo[]`

Ejemplos

```
float[] notas = new float[ALUMNOS];
int[] temperaturas = new int[7];
```

Matriz (array bidimensional):

```
matriz = new tipo[filas][columnas];
```

Ejemplo

```
int[][] temperaturas = new int[12][31];
```

Uso

Para acceder a los elementos de un array,
utilizamos índices
(para indicar la posición del elemento dentro del array)

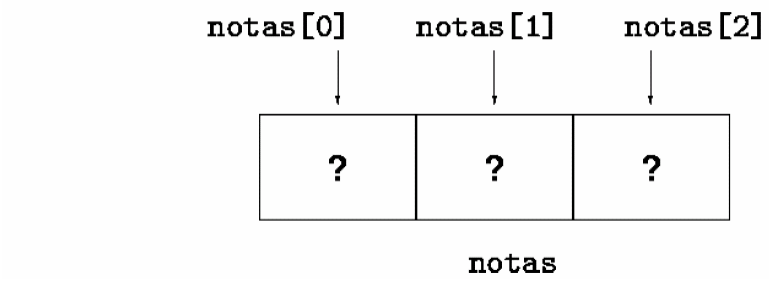
Vector (array unidimensional):

```
vector[índice]
```

- En Java, el índice de la primera componente de un vector es siempre 0.
- El tamaño del array puede obtenerse utilizando la propiedad `vector.length`
- Por tanto, el índice de la última componente es `vector.length-1`

Ejemplo

```
float[] notas = new float[3];
```



Matriz (array bidimensional):

```
matriz[índice1][índice2]
```

Una matriz, en realidad, es un vector de vectores:

- En Java, el índice de la primera componente de un vector es siempre 0, por lo que `matriz[0][0]` será el primer elemento de la matriz.
- El tamaño del array puede obtenerse utilizando la propiedad `array.length`:
  - `matriz.length` nos da el número de filas
  - `matriz[0].length` nos da el número de columnas
- Por tanto, el último elemento de la matriz es `matriz[matriz.length-1][matriz[0].length-1]`

Inicialización en la declaración

Podemos asignarle un valor inicial
a los elementos de un array en la propia declaración

```
int vector[] = {1, 2, 3, 5, 7};
int matriz[][] = { {1,2,3}, {4,5,6} };
```

El compilador deduce automáticamente las dimensiones del array.

Manipulación de vectores y matrices

Las operaciones se realizan componente a componente

Ejemplo: Suma de los elementos de un vector

```
static float media (float datos[])
{
  int   i;
  int   n = datos.length;
  float suma = 0;

  for (i=0; i<n; i++)
      suma = suma + datos[i];

  return suma/n;
}
```

No es necesario utilizar todos los elementos de un vector, por lo que, al trabajar con ellos, se puede utilizar una variable entera adicional que nos indique el número de datos que realmente estamos utilizando:

El tamaño del vector nos dice cuánta memoria se ha reservado para almacenar datos del mismo tipo, no cuántos datos del mismo tipo tenemos realmente en el vector.

Ejemplo: Suma de los n primeros elementos de un vector

```
static float media (float datos[], int n)
{
  int   i;
  float suma = 0;

  for (i=0; i<n; i++)
      suma = suma + datos[i];

  return suma/n;
}
```

Ejemplo

```
public class Vectores
{
  public static void main (String[] args)
  {
    int pares[] = { 2, 4, 6, 8, 10 };
    int impares[] = { 1, 3, 5, 7, 9 };

    mostrarVector(pares);
    System.out.println("MEDIA="+media(pares));

    mostrarVector(impares);
    System.out.println("MEDIA="+media(impares));
  }

  static void mostrarVector (int datos[])
  {
    int   i;

    for (i=0; i<datos.length; i++)
        System.out.println(datos[i]);
  }

  static float media (int datos[])
  {
    int i;
    int n = datos.length;
    int suma = 0;

    for (i=0; i<n; i++)
        suma = suma + datos[i];

    return suma/n;
  }
}
```

```
static int[] leerVector (int datos)
{
  int   i;
  int[] vector = new int[datos];

  for (i=0; i<datos; i++)
      vector[i] = leerValor();

  return vector;
}
```

IMPORTANTE:

Cuando se pasa un array como parámetro,
se copia una referencia al array y no el conjunto de valores en sí.

Por tanto, tenemos que tener cuidado con los efectos colaterales
que se producen si, dentro de un módulo,
modificamos un vector que recibimos como parámetro.

Ejemplo
El siguiente método lee los elementos de un vector ya creado

```
static void leerVector (int[] datos)
{
  int   i;

  for (i=0; i<datos.length; i++)
      datos[i] = leerValor();
}
```

Copia de arrays

La siguiente asignación sólo copia las referencias,
no crea un nuevo array:

```
int[] datos = pares;
```

Para copiar los elementos de un array,
hemos de crear un nuevo array y copiar los elementos uno a uno

```
int[] datos = new int[pares.length];

for (i=0; i<pares.length; i++)
    datos[i] = pares[i]
```

También podemos utilizar una función predefinida
en la biblioteca de clases estándar de Java:

```
System.arraycopy(from,fromIndex,to,toIndex,n);
```

```
int[] datos = new int[pares.length];
System.arraycopy(pares,0,datos,0,pares.length);
```

EXTRA:
La biblioteca de clases de Java incluye una clase auxiliar llamada **`java.util.Arrays`** que incluye como métodos algunas de las tareas que se realizan más a menudo con vectores:

- `Arrays.sort(v)` ordena los elementos del vector.
- `Arrays.equals(v1,v2)` comprueba si dos vectores son iguales.
- `Arrays.fill(v,val)` rellena el vector `v` con el valor `val`.
- `Arrays.toString(v)` devuelve una cadena que representa el contenido del vector.
- `Arrays.binarySearch(v, k)` busca el valor `k` dentro del vector `v` (que previamente ha de estar ordenado).

Ejemplos

Un programa que muestra los parámetros
que le indicamos en la línea de comandos:

```
public class Eco
{
  public static void main(String args[])
  {
    int i;

    for (i=0; i<args.length; i++)
        System.out.println(args[i]);
  }
}
```

Un método que muestra el contenido de una matriz:

```
public static void mostrarMatriz (double matriz[][])
{
  int i,j;
  int filas = matriz.length;
  int columnas = matriz[0].length;

  // Recorrido de las filas de la matriz

  for (i=0; i<filas; i++) {

      // Recorrido de las celdas de una fila

      for (j=0; j<columnas; j++) {

          System.out.println ( "matriz["+i+"]["+j+"]="
                               + matriz[i][j] );
      }
  }
}
```