Vectores o arreglos unidimensionales

Jonatan Goméz Perdomo, Ph.D.
jgomezpe@unal.edu.co

Arles Rodríguez, Ph.D.
aerodriguezp@unal.edu.co

Camilo Cubides, Ph.D.(c)
eccubidesg@unal.edu.co

Grupo de investigación en vida artificial – Research Group on Artificial Life – (Alife)
Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia

Agenda

Vectores I

Un *vector* es una *n*-tupla de *n* objetos llamados las *componentes* del vector, los cuales generalmente son números, caracteres, valores booleanos, y cualesquiera otro tipo de elementos que pertenecen a un mismo conjunto.

Ejemplo

La siguiente quíntupla de números enteros denotada por la expresión v es un vector

$$v = (1, 0, 7, -2, 8)$$

Vectores II

En general, un vector v se puede representar de la siguiente forma

$$v = (v_1, v_2, v_3, \ldots, v_n)$$

donde el vector está constituido por n componentes de un conjunto genérico V. Si $v \in V^n$, entonces el vector se dice que es n-dimensional o de tamaño n.

Ejemplo

El siguiente vector de tamaño 6 pertenece a $\mathbb{Z}^6$ y tiene una notación particular la cual es $\mathbf{0}_6$

$$\mathbf{0}_6 = (0, 0, 0, 0, 0, 0)$$

Vectores III

En un vector, para referirse a una componente en particular, a ésta se le dice que es la *componente* en la posición i o la i-ésima componente, esto significa que el objeto es la componente ubicada en la posición i, se denota por la expresión v_i y se puede ubicar dentro del vector como se muestra a continuación

componente i-ésima

$$\downarrow$$

$$(v_1, \ldots, v_i, \ldots, v_n)$$

Vectores IV

Ejemplo

Para el vector

$$v = \left(-1,\ \frac{3}{4},\ -0.25,\ -\frac{1}{5},\ \sqrt{2},\ 0.0,\ \pi,\ \sqrt[3]{5},\ 0.\overline{9} \right)$$

de $\mathbb{R}^9$ se tiene que sus componentes son:

- $v_1 = -1$.
- $v_2 = \frac{3}{4}$.
- $v_3 = -0.25$.
- $v_4 = -\frac{1}{5}$.
- $v_5 = \sqrt{2}$.
- $v_6 = 0.0$.
- $v_7 = \pi$.
- $v_8 = \sqrt[3]{5}$.
- $v_9 = 0.\overline{9}$.

El cojunto de los vectores

A partir de la notación de producto generalizado de un conjunto V

$$V^n = \underbrace{V \times V \times V \times \cdots \times V}_{n\text{-veces}}$$

se puede obtener el conjunto de los vectores $V[]$, el cual se define como la unión de todos los productos cartesianos del conjunto V, de la siguiente manera

$$V[] = \bigcup_{n \in \mathbb{N}} V^n$$

Agenda

Arreglos en computación

Si se tiene un conjunto $\mathbb{T}$ que representa un tipo de datos y un número $n \in \mathbb{N}$, se puede construir un vector de tamaño n, a los vectores que se construyen sobre tipos de datos en computación se les llama *arreglos* o *vectores unidimensionales*.
A partir del producto generalizado de un conjunto $\mathbb{T}$ se puede obtener el conjunto de los arreglos $\mathbb{T}[]$, el cual se define como la unión de todos los productos cartesianos del conjunto $\mathbb{T}$, de la siguiente manera

$$\mathbb{T}[] = \bigcup_{n \in \mathbb{N}} \mathbb{T}^n$$ y se llama el conjunto de todos los arreglos de tipo $\mathbb{T}$.

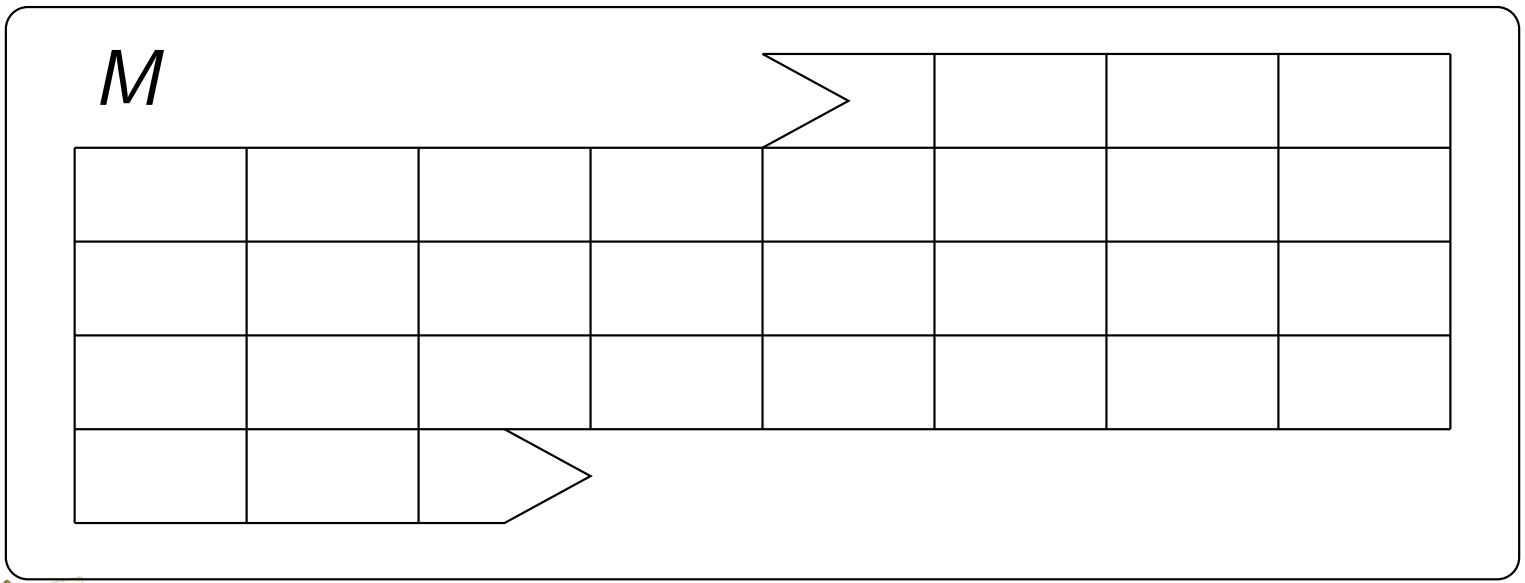
Arreglos en computación I

el conjunto de los arreglos del tipo de datos $\mathbb{T}$ es una colección de variables del tipo de datos $\mathbb{T}$.
Si se quiere expresar en Java que $\texttt{x} \in \mathbb{T}[]$ esto se escribe como `T[] x;` y para reservar el espacio de memoria para todo el arreglo de tipo $\mathbb{T}$, esto se escribe como `x = new T[n]`. De donde, para crear un arreglo `x` de tamaño n y de tipo $\mathbb{T}$ se utiliza la instrucción

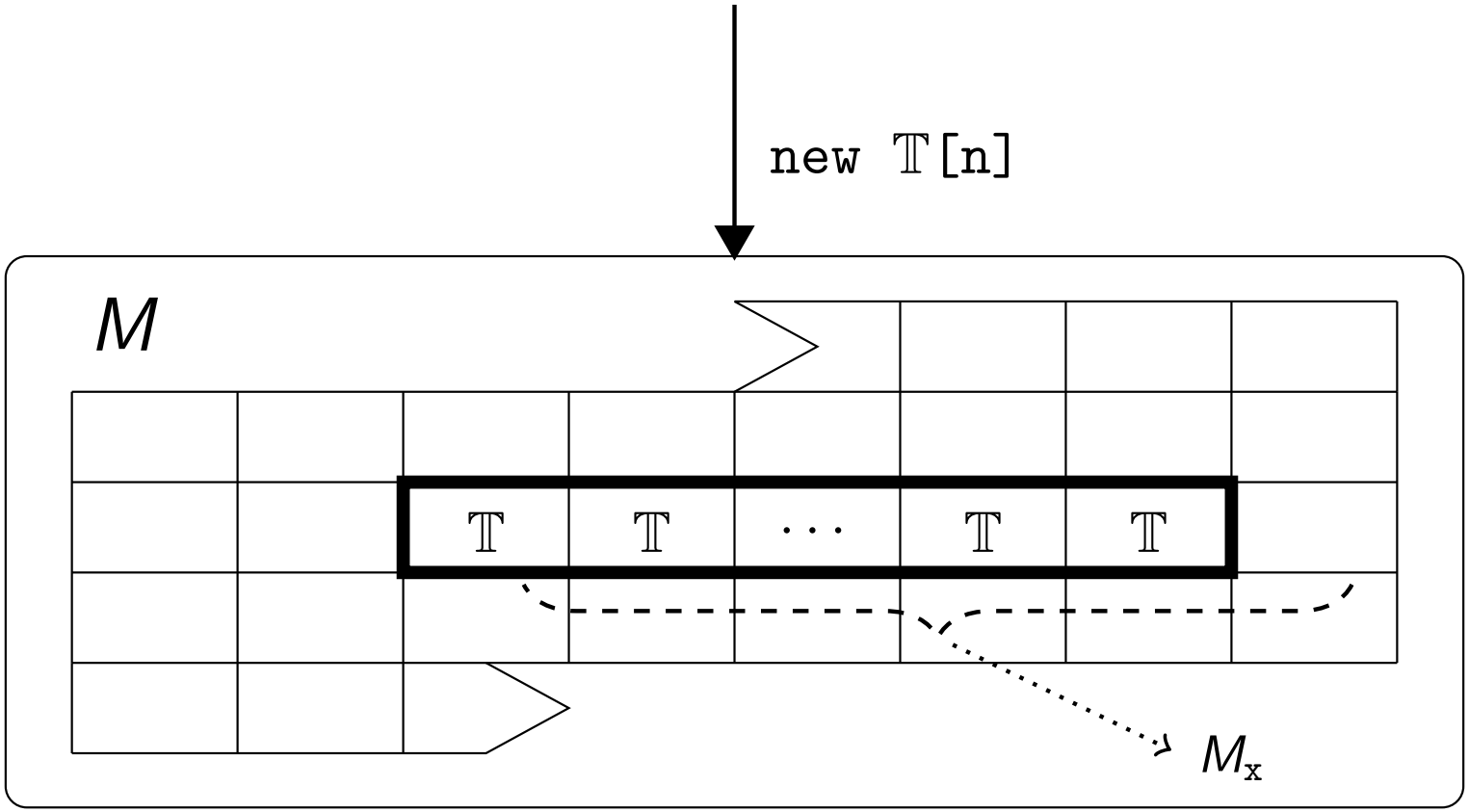
```
T[] x = new T[n];
```

Arreglos en computación II

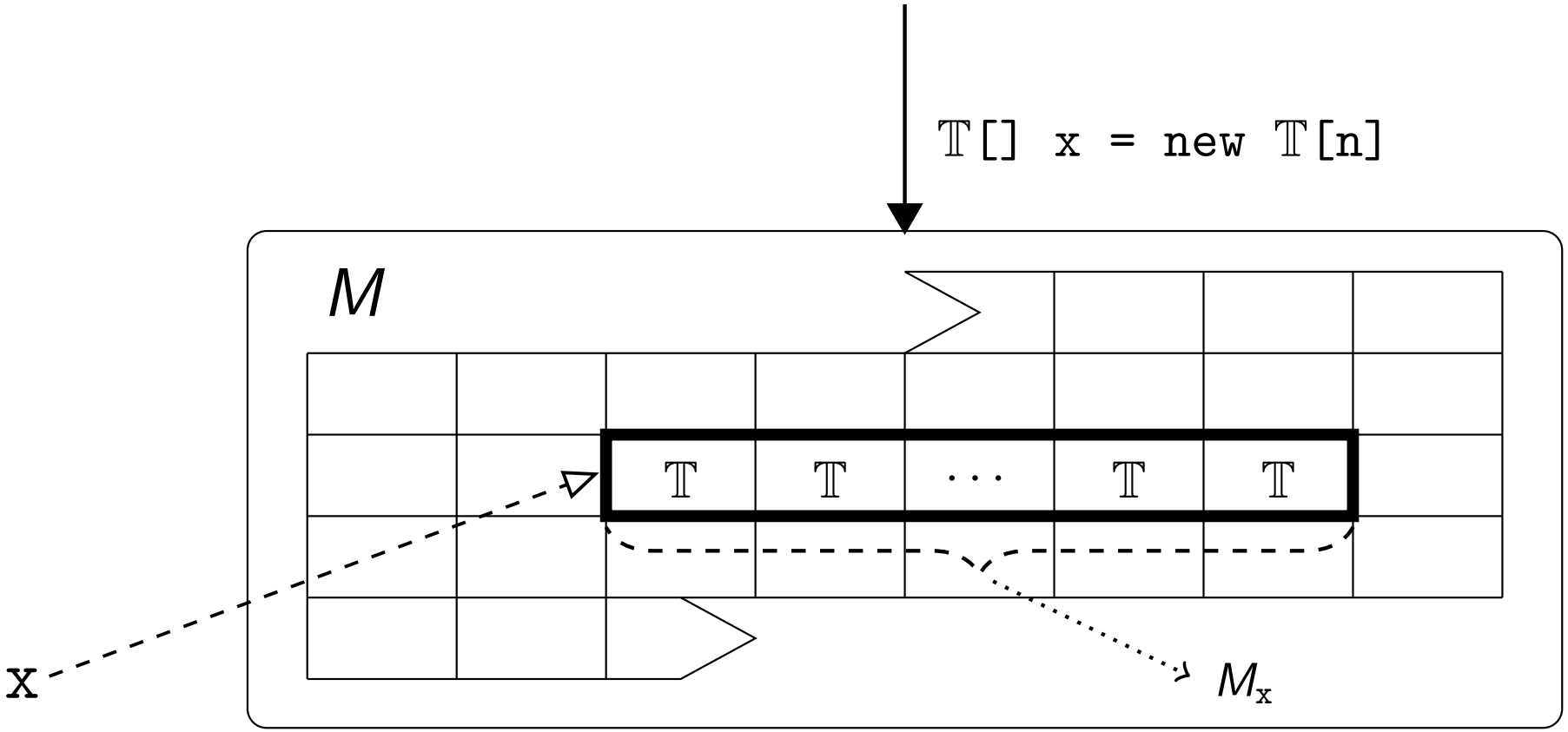
Con la instrucción $\mathbb{T}$`[] x = new` $\mathbb{T}$`[n]` se reserva una porción de la memoria M que es utilizada para almacenar el arreglo y que es bloqueada para que sólo se pueda utilizar para guardar valores en el arreglo. A la porción de espacio de memoria que ocupa un arreglo `x` lo notaremos como $M_{\texttt{x}}$. Gráficamente esto se puede ver así:



Arreglos en computación III



Arreglos en computación III



Arreglos en computación IV

Cuando se define un arreglo, lo que se construye es un conjunto de variables del mismo tipo, estas variables se encuentran subindicadas, esto es, que se accede a ellas por medio de un índice que específica una componente particular del arreglo.
En lenguajes como C++ y Java, es necesario tener en cuenta que la primera componente del arreglo está ubicada en la posición 0, y para un arreglo de tamaño n se tiene que la última componente del arreglo estará ubicada en la posición $n - 1$.

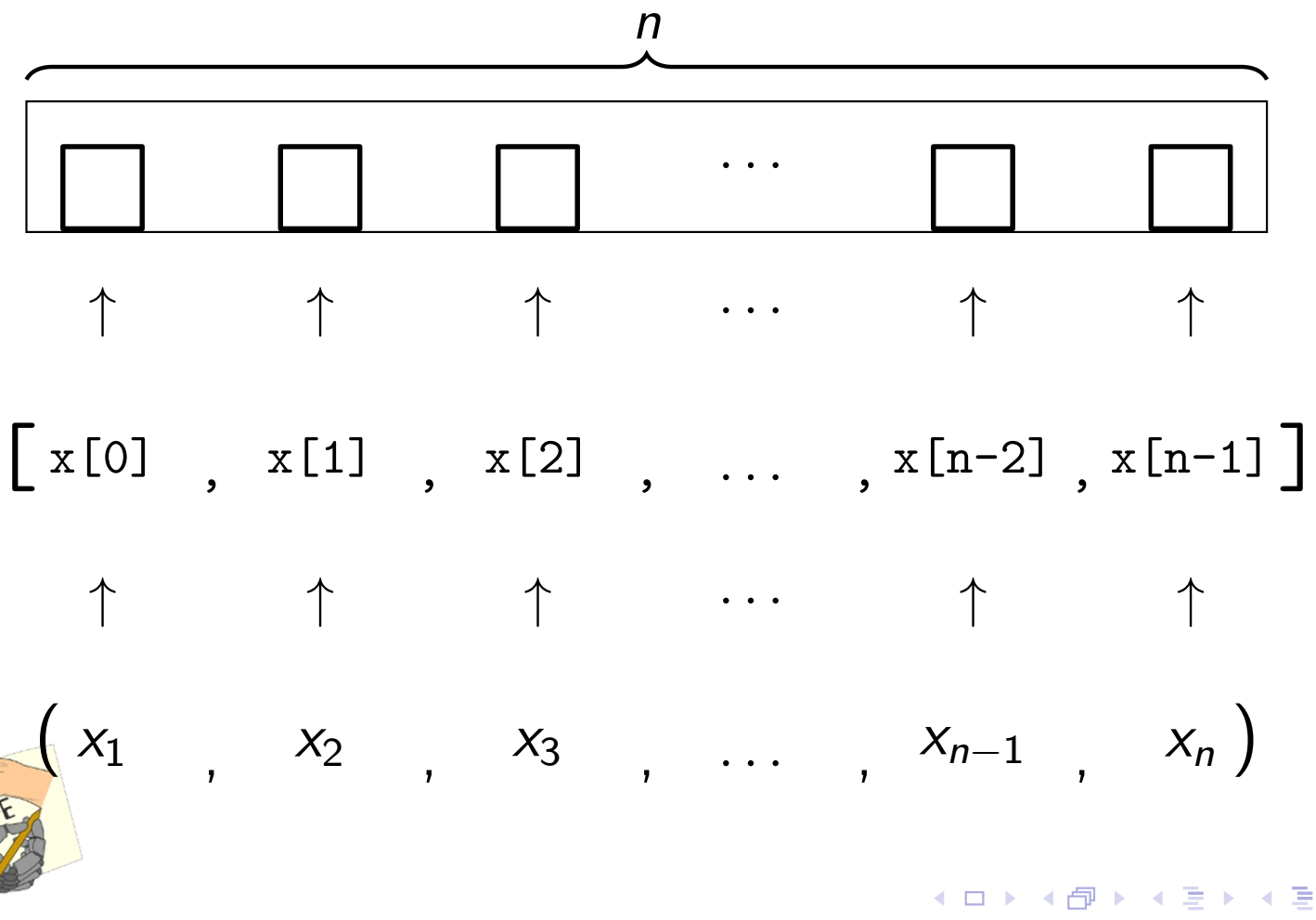
Arreglos en computación V

A partir de lo anterior se tiene que dado un arreglo $\mathtt{x} \in \mathbb{T}[]$, para acceder en C++ o Java a la variable almacenada en la componente *i* se utiliza la equivalencia

$$\mathtt{x}_i \equiv \mathtt{x[i-1]}$$

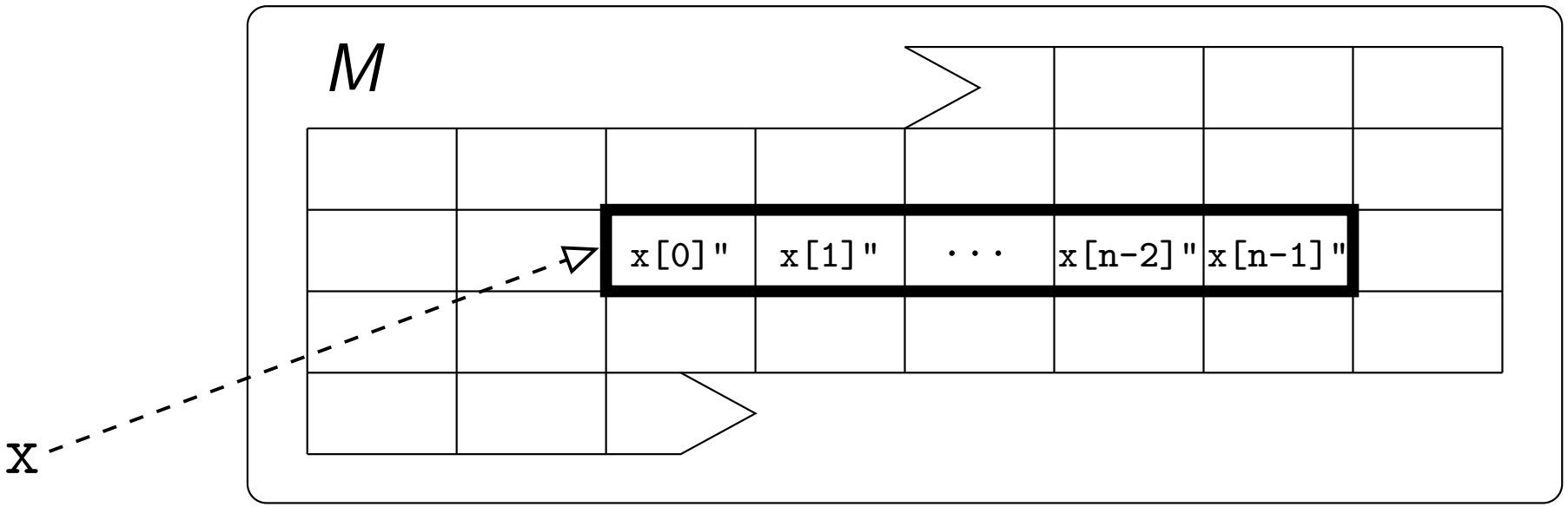
Arreglos en computación VI

es decir, para un vector $x = (x_1, x_2, x_3, \ldots, x_{n-1}, x_n)$ la representación del arreglo en Java es la que se presenta en la siguiente figura.



Arreglos en computación VII

con lo cual, la representación en la memoria resultará ser



Arreglos en computación VIII

En lenguajes como SciLab y MatLab, la primera componente del arreglo está ubicada en la posición 1, y para un arreglo de tamaño n se tiene que la última componente del arreglo estará ubicada en la posición n, por lo que las posiciones se manejan como en notación matemática de vectores, es decir, para acceder en SciLab o MatLab a la variable almacenada en la componente i se utiliza la equivalencia

$$\mathtt{x}_i \equiv \mathtt{x[i]}$$

Arreglos en computación IX

En caso de que se quiera acceder a una componente mayor o igual a la n-ésima en Java el programa será interrumpido mostrando al usuario una excepción de tipo `java.lang.ArrayIndexOutOfBoundsException`. Si se quiere acceder a una componente menor a 0, ocurre algo similar a lo descrito anteriormente.

Agenda

Creacion de arreglos I

Matemáticamente se definirá una rutina de creación de un arreglo como aquella rutina que dado un $n \in \mathbb{N}$, correspondiente al tamaño, retornará un elemento de $\mathbb{T}[]$ que es de tamaño n.

$$\begin{aligned} \mathit{crear\_arreglo\_}\mathbb{T} : \mathbb{N} &\to \mathbb{T}^* \\ (n) &\mapsto x, \quad \text{donde } x \in \mathbb{T}^n \subsetneq \mathbb{T}^* \end{aligned}$$

Creacion de arreglos II

En Java se traduce

```
T[] x = new T[n];
```

Ejemplo

Para crear un arreglo de tipo entero con nombre x se realiza el siguiente llamado:

```
int[] x = new int[n];
```

Eliminación de arreglos

La eliminación del espacio de memoria utilizado por el arreglo en Java es realizada por la máquina virtual una vez dicho espacio no esté asignado a una variable activa del programa. Siendo este un libro introductorio no se profundiza en detalles sobre el mecanismo de recolección de basura de Java conocido en inglés como *garbage collector*.

Agenda

flujos de entrada y salida

Dado un arreglo de tipo $\mathbb{T}$, es posible realizar operaciones de lectura y escritura. Una buena práctica de programación consiste en diseñar métodos para leer y escribir un arreglo en Java para empezar a diseñar código flexible y evitar la repetición de código cada vez que se quiera leer o escribir un arreglo en por consola. Los arreglos no son tipos primitivos así que no pueden leerse directamente utilizando `Scanner` ni pueden escribirse directamente utilizando `System.out.println`.

Agenda

flujos de entrada I

Lectura de arreglos: para la entrada o lectura de un arreglo desde la consola se requiere de un flujo de entrada en este caso el Scanner $\mathcal{SC}$ *sc*, del arreglo $\mathbb{T}[]$ *x* y de la cantidad de elementos a leer $\mathbb{N}$ *n*. Así se define la siguiente función

$$leerArreglo\mathbb{T} : \mathcal{SC} \times \mathbb{T}[] \times \mathbb{N} \to \mathbb{T}[]$$
$$(sc, x, n) \mapsto x, \quad \text{donde} \quad x_i = leer\_\mathbb{T}(is), \quad \forall i = 1, 2, 3, \ldots, n.$$

flujos de entrada II

La traducción de esta función en Java se escribe de la siguiente manera (es necesario tener en cuenta que como se mencionó previamente, los arreglos en Java comienzan en la posición 0):

```
public static 𝕋[] leerArreglo𝕋(Scanner sc, 𝕋[] x, int n){
   for(int i = 0; i < n; i++){
      x[i] = Class𝕋.parse𝕋(sc.nextLine());
   };
   return x;
};
```

flujos de entrada III

Ejemplo

En Java para leer un arreglo de tipo entero se utiliza la siguiente función

```
public static int[] leerArregloInt(Scanner sc,
                                   int[] x, int n) {
    for (int i = 0; i < n; i++) {
        x[i] = Integer.parseInt(sc.nextLine());
    }
    return x;
}
```

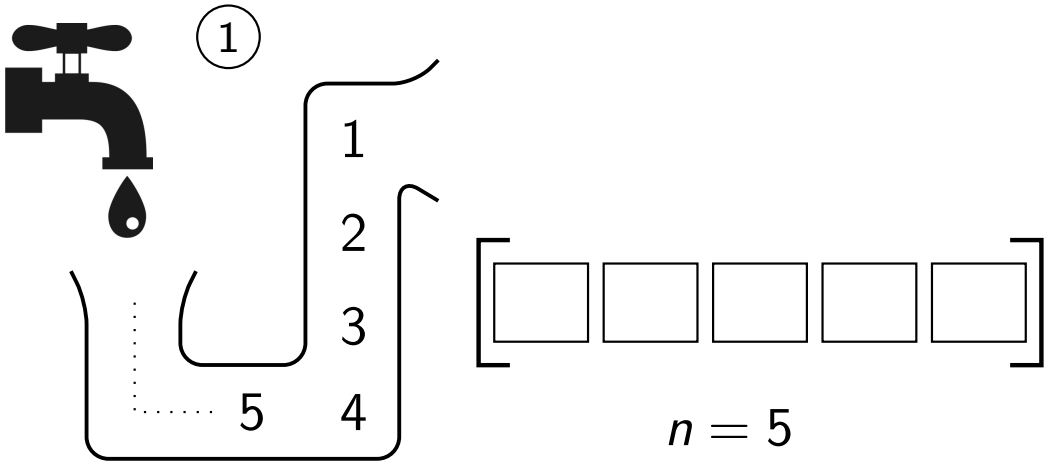
flujos de entrada IV

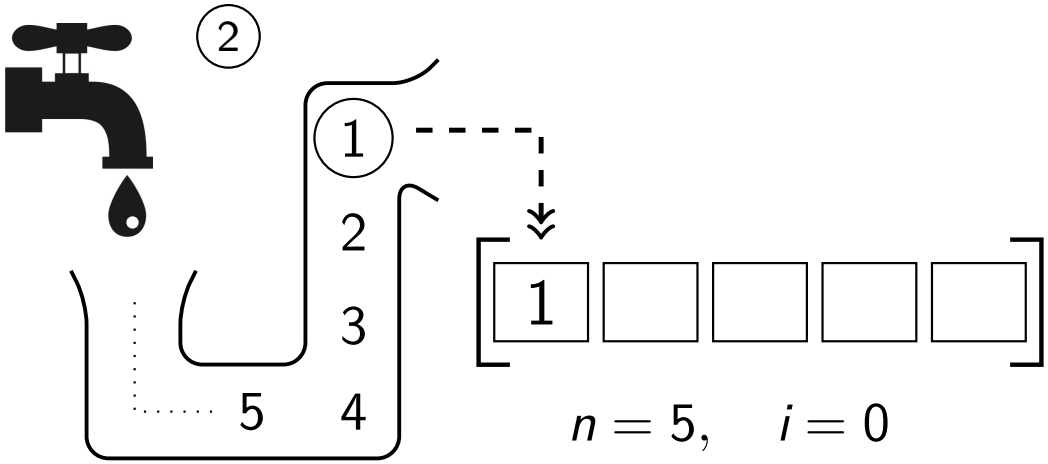
Otra opción para leer todos los elementos de un arreglo en Java puede ser utilizar la propiedad de tamaño del arreglo incluida en la definición de arreglos de Java. El tamaño de un arreglo `x` dado en Java puede obtenerse mediante la instrucción `x.length`.

Ejemplo

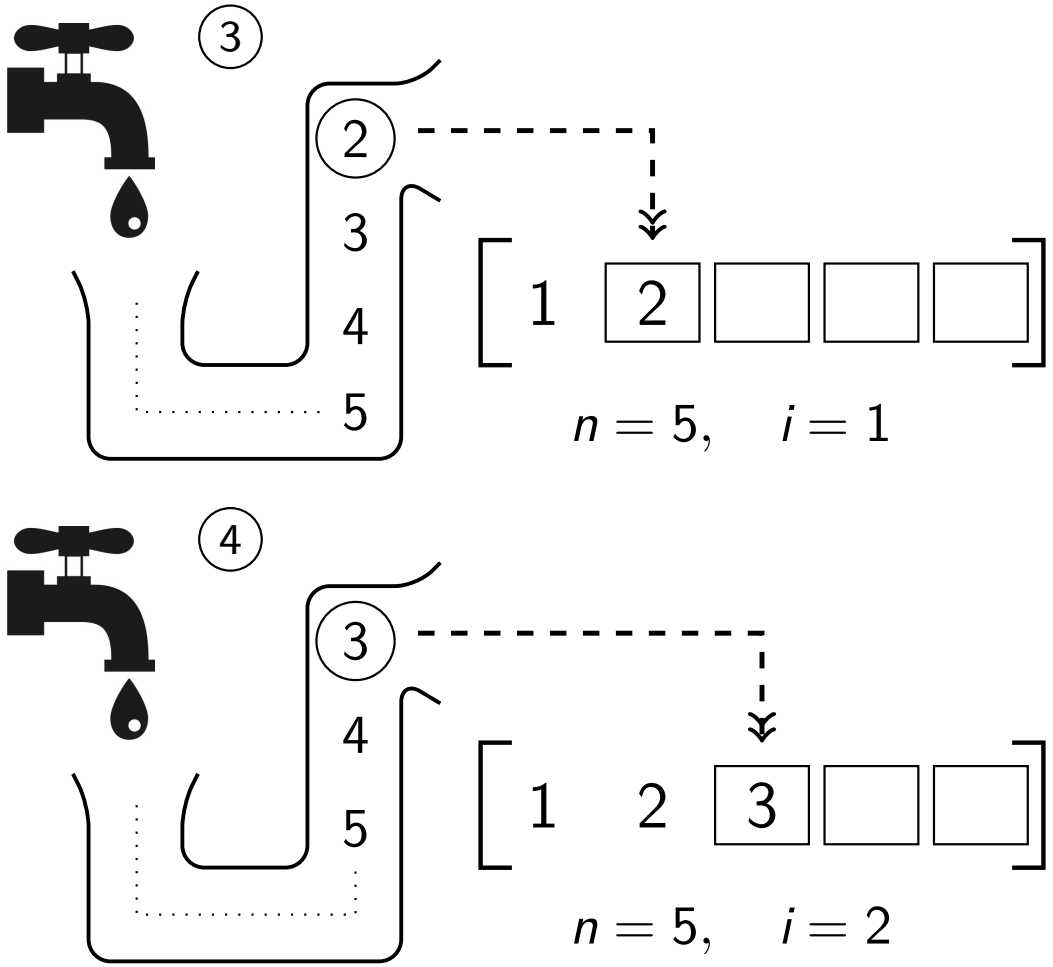
```
public static int[] leerArregloInt(Scanner sc, int[] x) {
    for (int i = 0; i < x.length; i++) {
        x[i] = Integer.parseInt(sc.nextLine());
    }
    return x;
}
```

flujos de entrada V

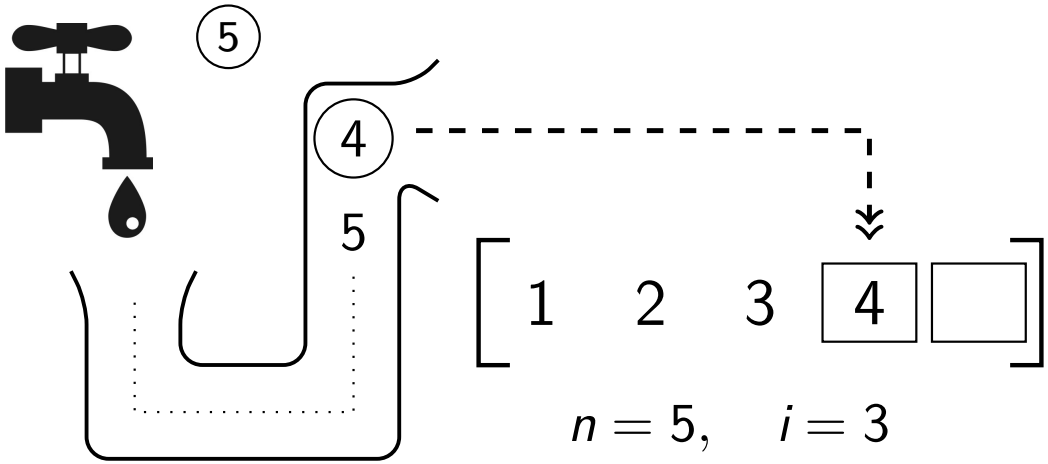


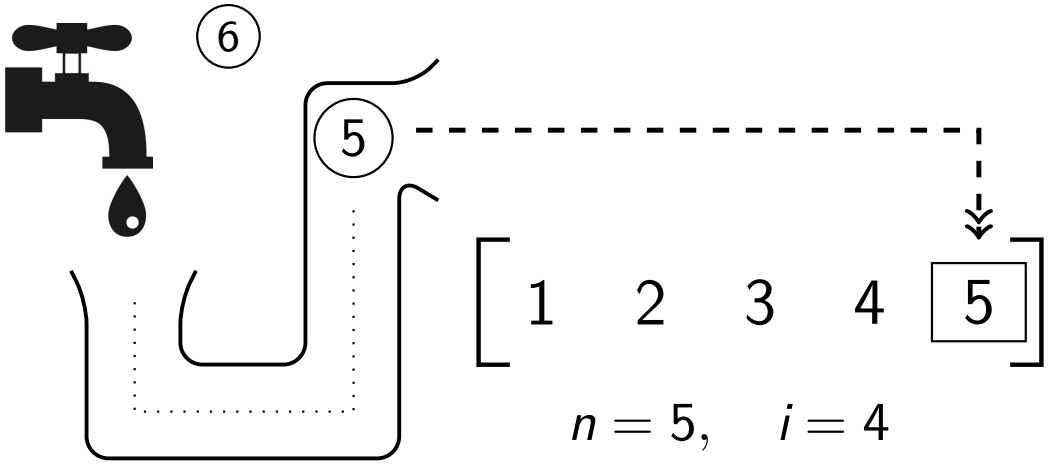


flujos de entrada VI

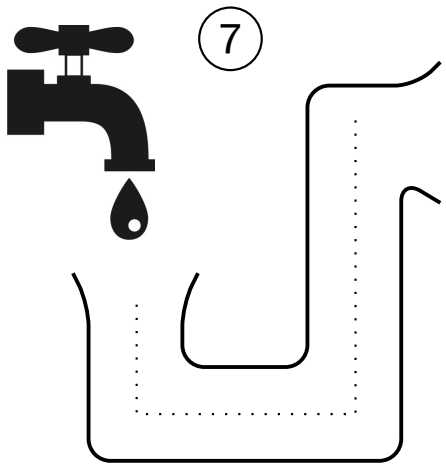


flujos de entrada VII





flujos de entrada VIII



$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

$$n = 5, \quad i = 5$$

Agenda

flujos de salida I

Escritura de arreglos: para escribir un arreglo en un la consola se define el siguiente procedimiento

$$\begin{aligned} escribirArreglo\mathbb{T} : \mathbb{T}[] \times \mathbb{N} &\to \emptyset \\ (x, n) &\mapsto \quad \text{donde} \quad escribir\mathbb{T}(x_i), \\ & \qquad\qquad \forall i = 1, 2, 3, \ldots, n. \end{aligned}$$

flujos de salida II

La traducción de este procedimiento en Java se escribe de la siguiente manera. Se escribe en el flujo de datos cada una de las componentes del arreglo separadas por un símbolo de tabulación o un espacio en blanco, esto con el fin de diferenciar las componentes entre sí. Al final se agrega una línea en blanco para separar el arreglo impreso de otras salidas que se quieran imprimir en la consola.

```
public static 𝕋[] leerArreglo𝕋(Scanner sc, 𝕋[] x, int n){
   for(int i = 0; i < n; i++){
      x[i] = Class𝕋.parse𝕋(sc.nextLine());
   }
   return x;
}
```

flujos de salida III

Ejemplo

En Java para escribir un arreglo de tipo entero se utiliza el siguiente procedimiento

```
public static void escribirArregloInt(int[] x, int n){
    for(int i = 0; i < n; i++){
        System.out.print(x[i] + " ");
    }
    System.out.println();
}
```

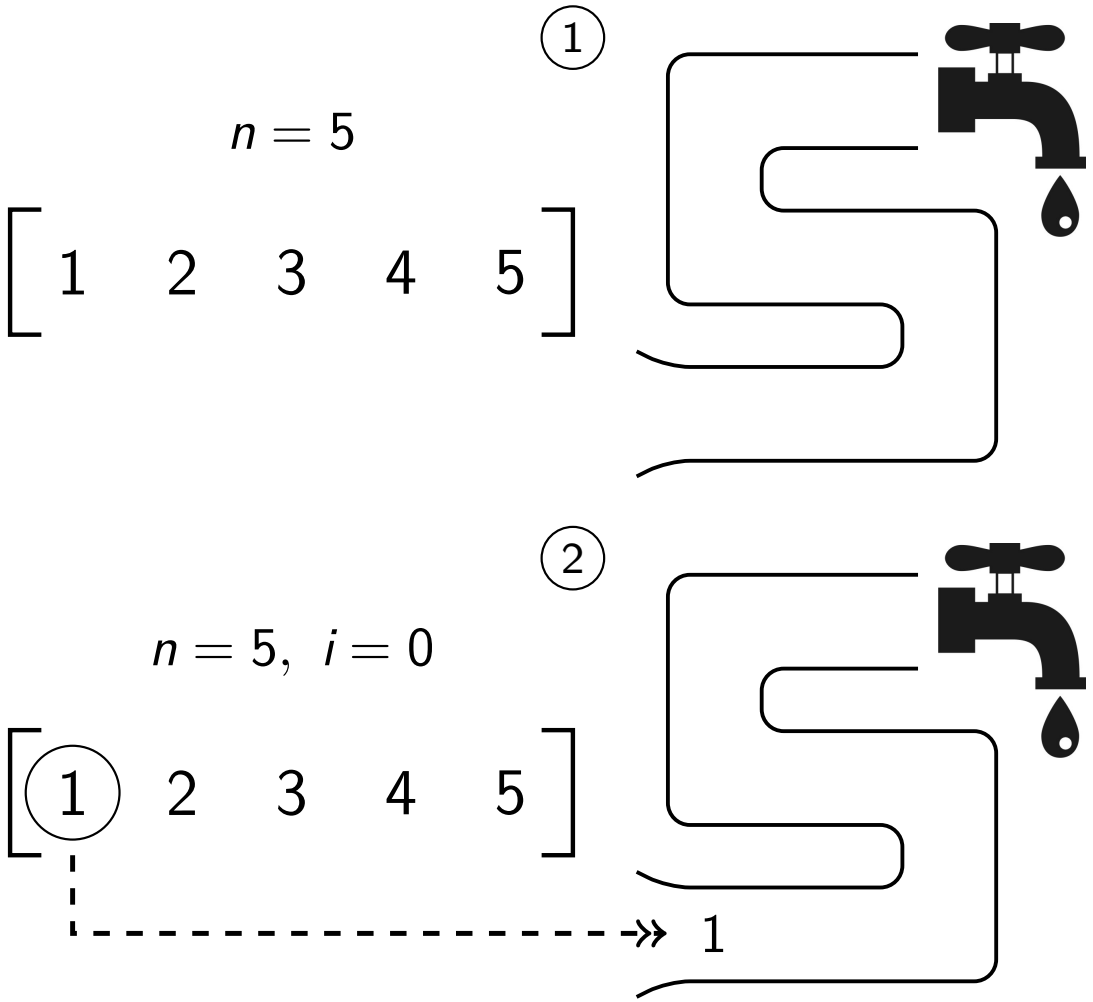
flujos de salida IV

Se podría utilizar también el siguiente procedimiento

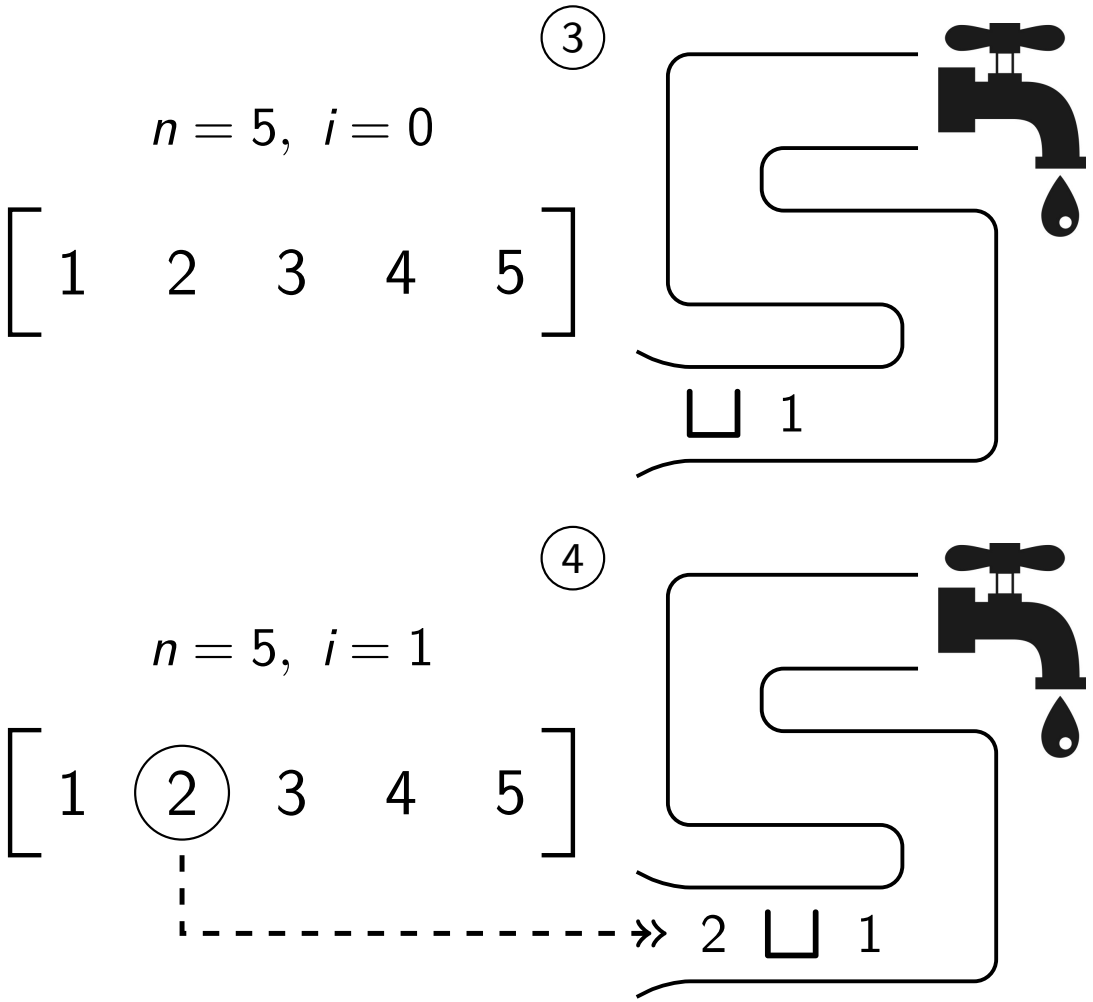
Ejemplo

```
public static void escribirArregloInt(int[] x){
    for (int i = 0; i < x.length; i++) {
        System.out.print(x[i] + " ");
    }
    System.out.println();
}
```

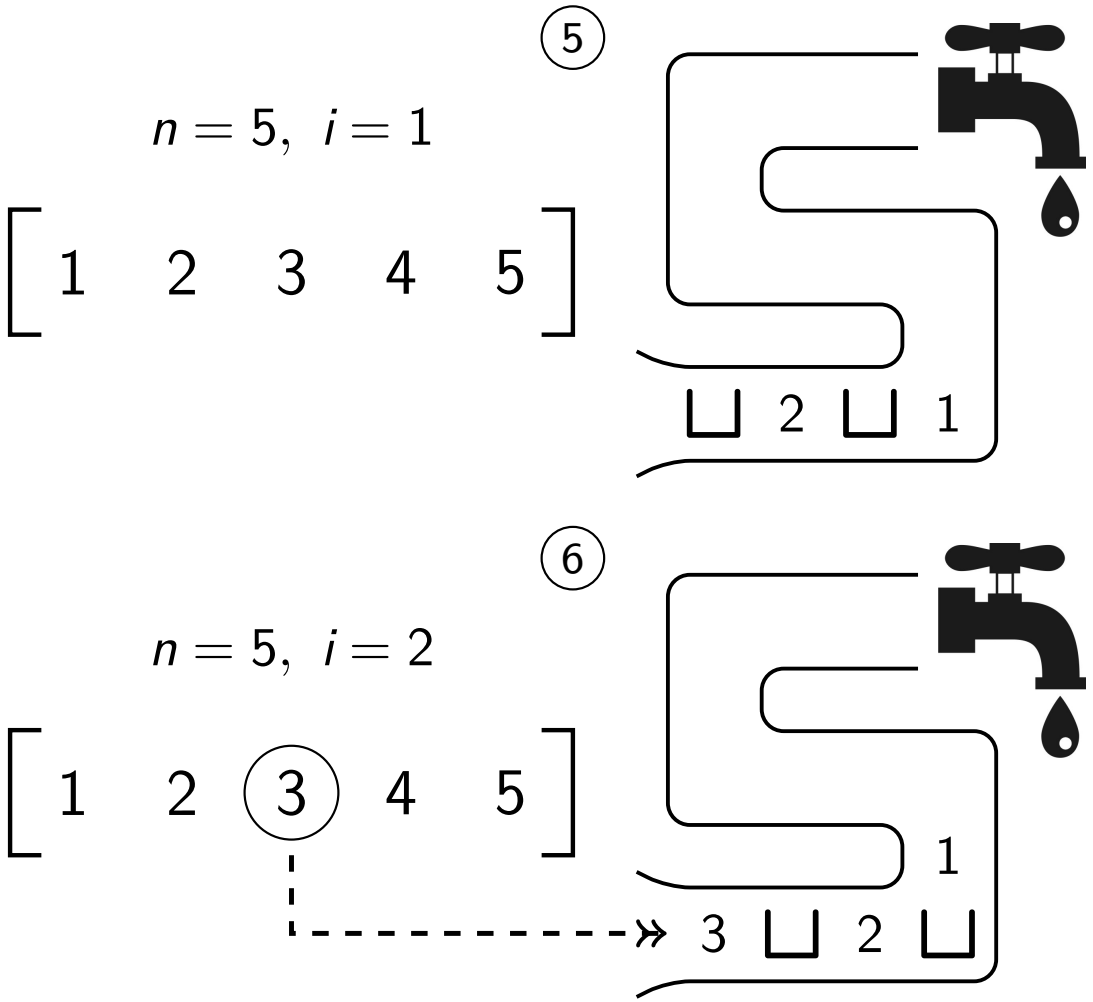
flujos de salida V



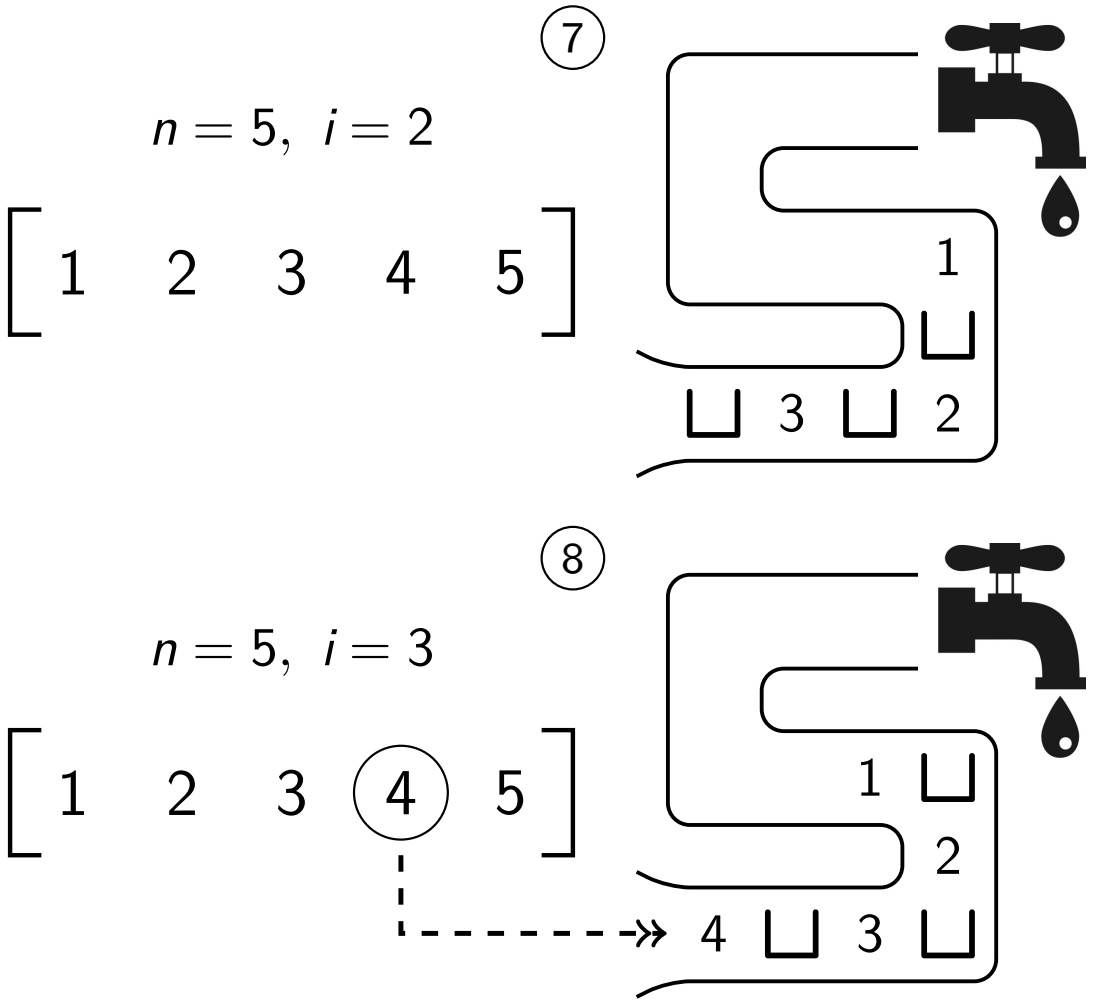
flujos de salida VI



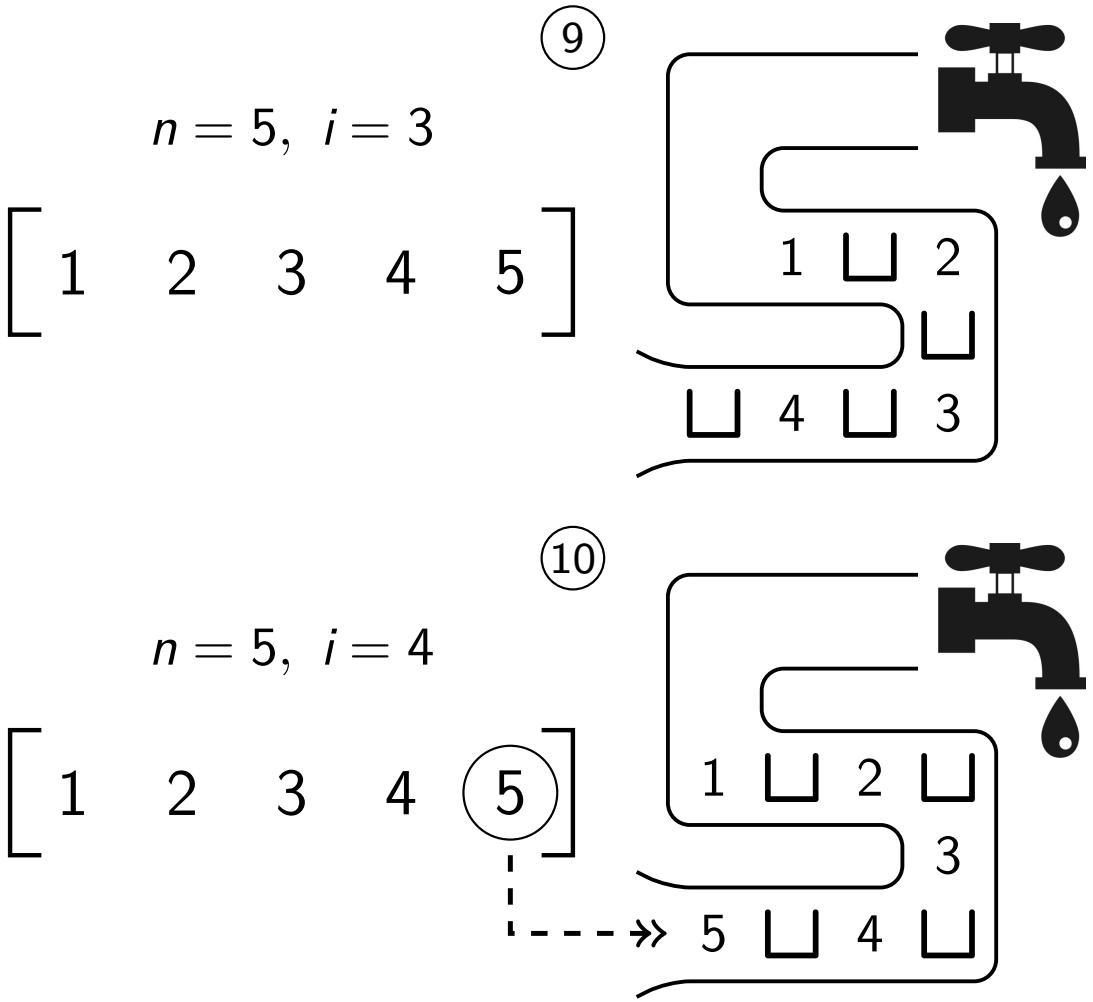
flujos de salida VII



flujos de salida VII



flujos de salida IX

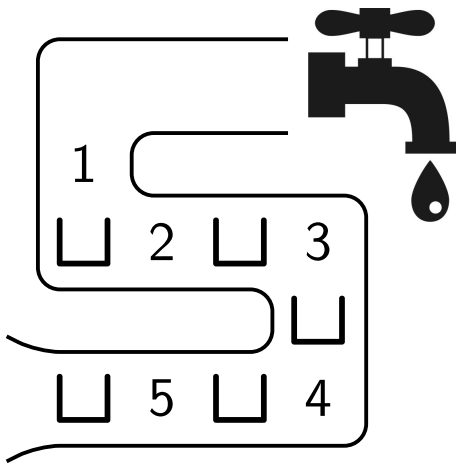


flujos de salida X

(11)

$n = 5,\ i = 4$

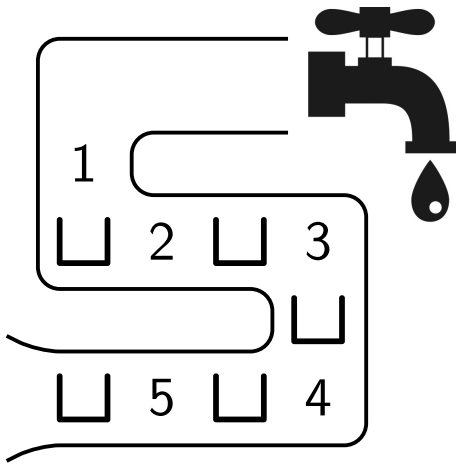
$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$



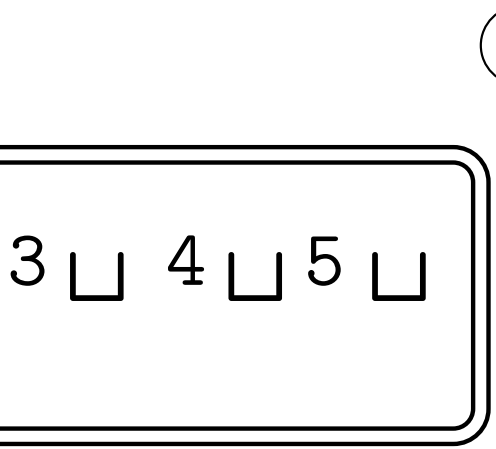
(12)

$n = 5,\ i = 5$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$



flujos de salida XI



Agenda

Ejemplo de funciones con arreglos I

Es posible utilizar lo visto en funciones para realizar diversidad de cálculos que involucren arreglos.

Ejemplo

El cubo de las componentes de arreglos numéricos enteros
Suponga que un arreglo contiene unos datos numéricos enteros tales que consta de 6 datos que se encuentran separados por espacio así como se muestra a continuación

```
1 2 3 4 5 6
```

Ejemplo de funciones con arreglos II

Una función general que permite construir un nuevo arreglo que contiene el cubo de cada componente de un arreglo dado es

$$\textit{cuboComponentesArreglo} : \mathbb{Z}[] \times \mathbb{N} \to \mathbb{Z}[]$$
$$(x, n) \mapsto y, \qquad \text{donde} \quad y_i = x_i^3, \quad \forall i = 1, 2, \ldots, n.$$

Ejemplo de funciones con arreglos III

Un programa completo en Java que permite calcular el cubo de las componentes de un arreglo obtenido a partir del archivo presentado anteriormente es:

Ejemplo

```
import java.util.Scanner;

public class ArregloInt {
```

```
    public static int[] leerArregloInt(Scanner sc, int[] x) {
        for (int i = 0; i < x.length; i++) {
            x[i] = sc.nextInt();
        }
        return x;
    }
```

Ejemplo de funciones con arreglos IV

Ejemplo

```
public static void escribirArregloInt(int[] x, int n) {
    for (int i = 0; i < x.length; i++) {
        System.out.print(x[i] + " ");
    }
    System.out.println();
}

public static int[] cuboComponentesArreglo(int[] x, int n) {
    int[] y = new int[n];
    for (int i = 0; i < n; i++) {
        y[i] = x[i] * x[i] * x[i];
    }
    return y;
}
```

Ejemplo de funciones con arreglos V

Ejemplo

```
public static void main(String[] args) {
    int n;
    Scanner sc = new Scanner(System.in);
    System.out.println("Digite el tamaño del"
            +" arreglo:");
    n = sc.nextInt();
    int[] x = new int[n];

    System.out.println("Digite los datos del arreglo"
            + " separados por espacio o enter:");
```

Ejemplos de funciones con arreglos VI

Ejemplo

```
        x = leerArregloInt(sc, x);
        int[] y = cuboComponentesArreglo(x, n);

        System.out.println("El arreglo elevado al cubo"
                +" es:");
        escribirArregloInt(y, n);
    }
}
```

Ejemplo de funciones con arreglos VII

La salida del programa tiene el siguiente aspecto

```
Digite el tamaño del arreglo:
6
Digite los datos del arreglo separados por espacio o enter:
1 2 3 4 5 6
El arreglo elevado al cubo es:
1 8 27 64 125 216
```

Problemas varios I

Problemas

1. Modele mediante una función matemática y desarrolle un algoritmo en Java con ciclos que permita hallar la posición del mínimo de un vector de números reales.
2. Modele mediante una función matemática y desarrolle un algoritmo con ciclos en Java que calcule el mínimo de un vector de números reales.

Problemas varios II

Problemas

4. Modele mediante una función matemática y desarrolle un algoritmo en Java que calcule el producto por escalar de una constante con un vector de números reales. Sea $\alpha \in \mathbb{R}$ y $v = (v_1, v_2, \ldots, v_n)$, el producto por escalar de α con v (notado αv) es el vector dado por la expresión

$$\alpha v = (\alpha v_1, \alpha v_2, \ldots, \alpha v_n)$$