



# Programación 1

Arreglos

# Arreglos

- Estructura de datos que contiene elementos o celdas del mismo tipo de dato.
- Son estáticos ya que su tamaño no cambia durante la ejecución
- Ocupa un serie de posiciones contiguas de memoria
- Cada celda tiene asociado un número entero que indica la posición respecto de las demás



## POSICIONES→

```
int A = vec[0] + vec[8];    // A = 12 + 6 = 18
int B = 2 + vec[3];        // B = 2 + 8 = 10
vec[0] = A + B;            // vec[0] = 18 + 10 = 28
```

# Declaración de arreglos

Arreglo nulo sin espacio para datos

```
int[] arrDatos;
```

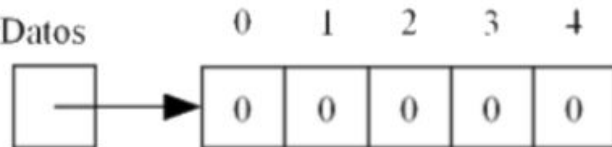
arrDatos



Arreglo inicializado con ceros

```
int[] arrDatos = new int[5];
```

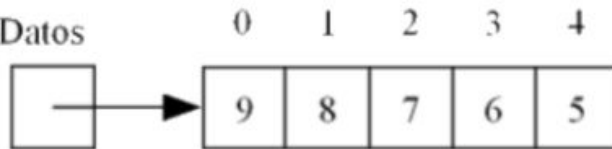
arrDatos



Arreglo inicializado con datos explícitos

```
int[] arrDatos = {9, 8, 7, 6, 5};
```

arrDatos



# Acceso a un arreglo



Una vez que el arreglo tiene espacio asignado el paso siguiente es inicializarlo con valores o cargarlo desde el teclado. Para acceder a cada espacio del arreglo se utiliza un valor de posición o variable con dicho valor, que va desde **0** a la **CANTIDAD-1**. Por ejemplo:

```
arrDatos[0] = 1;  
//se accede a la posición 0 del arreglo y se le asigna un 1  
pos = 2; //suponiendo que pos es un entero  
arreDatos[pos] = 5;  
//se accede a la posición pos del arreglo y se le asigna un 5, pos tiene que tener un valor entre 0 y  
//CANTIDAD-1 de elementos del arreglo, sino dará ERROR
```

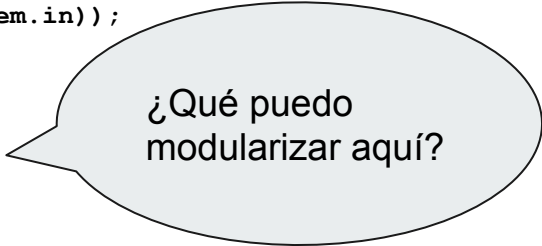
El acceso a una posición mediante un valor no se recomienda, siempre se utiliza una variable entera cuyo valor va de **0** a la **CANTIDAD-1**.

# Carga de un arreglo

//Hacer un programa que cargue en un arreglo de enteros 5 valores desde teclado y lo imprima.

//ESTA HECHO SIN METODOS, SOLO PARA EXPLICAR (MAS ADELANTE SE HACE CON METODOS)

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_5_Ejemplo_1 {
    public static void main (String [] args) {
        final int MAX = 5;
        int [] arrenteros = new int [MAX];
        BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
        try{
            for (int pos = 0; pos < MAX; pos++){
                System.out.println("Ingrese un entero: ");
                //arrenteros es un arreglo, arrenteros[pos] es como una
                //variable de tipo entero a la que se le asigna un valor
                arrenteros[pos] = Integer.valueOf(entrada.readLine());
            }
            for (int pos = 0; pos < MAX; pos++){
                System.out.println("arrenteros["+pos+"] -> "+arrenteros[pos]);
            }
        }
        catch(Exception exc){
            System.out.println(exc);
        }
    }
}
```



¿Qué puedo modularizar aquí?

# Carga de un arreglo (modularizado)

```
public class Clase_5_Ejemplo_1{
    final static int MAX = 5;
    public static void main (String[] args) {
        int[] enteros=new int[MAX];
        cargarArreglo(enteros);
        mostrarArreglo(enteros);
    }
    public static void cargarArreglo(int[] arrenteros){
        BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
        try{
            for (int pos = 0; pos < MAX; pos++){
                System.out.println ("Ingrese el entero de la posición " + pos + ": ");
                arrenteros[pos] = Integer.valueOf(entrada.readLine());
            }
        }
        catch(Exception exc){
            System.out.println(exc);
        }
    }
}
```

Sigue

```
public static void mostrarArreglo(int[] arrenteros){
    for (int pos = 0; pos < MAX; pos++){
        System.out.println("arrenteros["+pos+"] -> "+arrenteros[pos]);
    }
}
```

¿Qué pasa si no  
ingresa un entero?

# Carga de un arreglo - Consideraciones



Con el fin de simplificar los ejercicios, vamos a suponer que siempre nos dan arreglos cargados con datos. Para hacer pruebas de cómo funciona un código pueden usar un método que cargue desde consola. O podrán optar por cargas aleatorias. En última instancia pueden declarar por extensión con datos.



# Carga de un arreglo de forma aleatoria

```
import java.util.Random;

public class Clase_5_Ejemplo_2 {
    public static final int MAX = 10;
    public static final int MAXVALOR = 10;
    public static final int MINVALOR = 1;
    public static void main(String[] args) {
        char [] arrchar = new char[MAX];
        int [] arrint = new int[MAX];
        double [] arrdouble = new double[MAX];
        cargar_arreglo_aleatorio_char(arrchar);
        cargar_arreglo_aleatorio_int(arrint);
        cargar_arreglo_aleatorio_double(arrdouble);
        imprimir_arreglo_char(arrchar);
        imprimir_arreglo_int(arrint);
        imprimir_arreglo_double(arrdouble);
    }

    //carga de arreglo de char con valores de 'a' a la 'z'
    public static void cargar_arreglo_aleatorio_char(char [] arr){
        Random r = new Random();
        for (int pos = 0; pos < MAX; pos++){
            arr[pos]=(char) (r.nextInt(26) + 'a');
        }
    }

    //carga de arreglo de int con valores de MINVALOR a MAXVALOR
    public static void cargar_arreglo_aleatorio_int(int [] arr){
        Random r = new Random();
        for (int pos = 0; pos < MAX; pos++){
            arr[pos]=(r.nextInt(MAXVALOR-MINVALOR+1) + MINVALOR);
        }
    }

    //carga de arreglo de double con valores de MINVALOR a MAXVALOR
    public static void cargar_arreglo_aleatorio_double(double [] arr){
        Random r = new Random();
        for (int pos = 0; pos < MAX; pos++){
            arr[pos]=((MAXVALOR-MINVALOR+1)*r.nextDouble() +
            MINVALOR*1.0);
        }
    }

    //impresion de arreglo de char
    public static void imprimir_arreglo_char(char [] arr){
        for (int pos = 0; pos < MAX; pos++){
            System.out.println("nombre_arreglo["+pos+"]=>: "+arr[pos]);
        }
    }

    //impresion de arreglo de int
    public static void imprimir_arreglo_int(int [] arr){
        for (int pos = 0; pos < MAX; pos++){
            System.out.println("nombre_arreglo["+pos+"]=>: "+arr[pos]);
        }
    }

    //impresion de arreglo de double
    public static void imprimir_arreglo_double(double [] arr){
        for (int pos = 0; pos < MAX; pos++){
            System.out.println("nombre_arreglo["+pos+"]=>: "+arr[pos]);
        }
    }
}
```

# Ejemplo



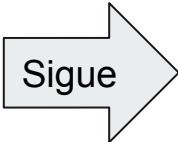
//Hacer un programa que dado un arreglo de enteros de tamaño 10 que se encuentra  
//precargado, imprima por pantalla el promedio de la suma de sus valores.

```
public class Clase_5_Ejemplo_3 {  
    public static int MAX = 10;  
    public static void main (String [] args) {  
        int [] arrenteros = new int [MAX];  
        int promedio;  
        //cargar el arreglo con alguno de los métodos propuestos  
        cargar_arreglo_int(arrenteros);  
        promedio = promedio_arreglo(arrenteros);  
        System.out.println("El promedio del arreglo es: " + promedio);  
    }  
    public static int promedio_arreglo(int [] arr){  
        int suma = 0;  
        for (int pos = 0; pos < MAX; pos++){  
            suma+=arr[pos];  
        }  
        return (suma/MAX);  
    }  
}
```

# Ejemplo

//Hacer un programa que dado un arreglo de enteros de tamaño 10 que se encuentra precargado, encuentre la posición de un número entero ingresado por el usuario. Si existe, muestre esa posición por pantalla, o indique que no existe.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_5_Ejemplo_4 {
    public static int MAX = 10;
    public static void main (String [] args) {
        int [] arrenteros = new int [MAX];
        int pos,numero;
        BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
        try{
            cargar_arreglo_aleatorio_int(arrenteros);
            imprimir_arreglo_int(arrenteros);
            System.out.println("Ingrese un numero entero :");
            numero = Integer.valueOf(entrada.readLine());
            pos = obtener_pos_arreglo(arrenteros,numero);
            if (pos<MAX){
                System.out.println(numero + " está en " + pos);
            }
            else{
                System.out.println("No existe " + numero);
            }
        }
        catch(Exception exc){
            System.out.println(exc);
        }
    }
}
```



Sigue

```
public static int obtener_pos_arreglo(int [] arr, int numero){
    int posicion = 0;
    while ((posicion < MAX) && (arr[posicion] != numero)){
        posicion++;
    }
    return posicion;
}
```

# Pasaje de parámetros



Por copia valor:

- Cuando se invoca al método se crea una nueva variable (el parámetro formal) y se le copia el valor del parámetro actual.
- El parámetro actual y el formal son dos variables distintas aunque puedan llegar a tener el mismo nombre.
- El método trabaja con la copia de la variable por lo que cualquier modificación que se realice sobre ella dentro del método no afectará al valor de la variable afuera.

Por referencia:

- Cuando se invoca al método se crea una nueva variable (el parámetro formal) a la que se le asigna la dirección de memoria donde se encuentra el parámetro actual.
- En este caso el método trabaja con la variable original por lo que puede modificar su valor.

# Pasaje de parámetros



Por copia valor:

- Cuando se invoca al método se crea una nueva variable (el parámetro formal) y se le copia el valor del parámetro actual.
- El parámetro actual y el formal son dos variables diferentes que pueden llegar a tener el mismo nombre.
- El método trabaja con la copia de la variable por lo que cualquier modificación que se realice sobre ella dentro del método no afectará al valor de la variable afuera.



Ya visto en diseño descendente

Por referencia:

- Cuando se invoca al método se crea una nueva variable (el parámetro formal) a la que se le asigna la dirección de memoria donde se encuentra el parámetro actual.
- En este caso el método trabaja con la variable original por lo que puede modificar su valor.

# Pasaje de parámetros en java



En Java todos los parámetros se pasan por copia valor.

- Cuando el argumento es de tipo primitivo, el paso por copia valor significa que cuando se invoca al método se reserva un nuevo espacio en memoria para el parámetro formal. El método no puede modificar el parámetro actual.
- Cuando el argumento es una referencia (por ejemplo, un array o cualquier otro objeto) el paso por copia valor significa que el método recibe una copia de la dirección de memoria donde se encuentra el objeto. Por lo tanto el método puede modificar el contenido.

# Ejemplo

```
public class Clase_5_Ejemplo_4 {  
    final static int MAX = 5;  
    public static void main (String[] args) {  
        int[] B = new int[MAX];  
        int a = 20;  
        cargar_arreglo(B);  
        System.out.println("Los datos son:");  
        for (int pos = 0 ; pos < MAX; pos++)  
            System.out.println(B[pos]);  
        cargar_variable_simple(a);  
        System.out.println("La variable es:"+a);  
    }  
    private static void cargar_variable_simple(int c) {  
        c = 10;  
    }  
    private static void cargar_arreglo(int[] arr) {  
        for (int pos = 0 ; pos < MAX; pos++)  
            arr[pos]=pos*2;  
    }  
}
```


# Ejemplo

```
public class Clase_5_Ejemplo_4 {
    final static int MAX = 5;

    public static void main (String[] args) {
        int[] B = new int[MAX];
        int a = 20;
        cargar_arreglo(B);
        System.out.println("Los datos son:");
        for (int pos = 0 ; pos < MAX; pos++)
            System.out.println(B[pos]);
        cargar_variable_simple(a);
        System.out.println("La variable es:"+a);
    }
    private static void cargar_variable_simple(int c) {
        c = 10;
    }
    private static void cargar_arreglo(int[] arr) {
        for (int pos = 0 ; pos < MAX; pos++)
            arr[pos]=pos*2;
    }
}
```




# Ejemplo



```
public class Clase_5_Ejemplo_4 {
    final static int MAX = 5;

    public static void main (String[] args) {
        int[] B = new int[MAX];
        int a = 20;
        cargar_arreglo(B);
        System.out.println("Los datos son:");
        for (int pos = 0 ; pos < MAX; pos++)
            System.out.println(B[pos]);
        cargar_variable_simple(a);
        System.out.println("La variable es:"+a);
    }
    private static void cargar_variable_simple(int c) {
        c = 10;
    }
    private static void cargar_arreglo(int[] arr) {
        for (int pos = 0 ; pos < MAX; pos++)
            arr[pos]=pos*2;
    }
}
```


# Ejemplo



```
public class Clase_5_Ejemplo_4 {
    final static int MAX = 5;

    public static void main (String[] args) {
        int[] B=new int[MAX];
        int a = 20;
        cargar_arreglo(B);
        System.out.println("Los datos son:");
        for (int pos = 0 ; pos < MAX; pos++)
            System.out.println(B[pos]);
        cargar_variable_simple(a);
        System.out.println("La variable es:"+a);
    }
    private static void cargar_variable_simple(int c) {
        c = 10;
    }
    private static void cargar_arreglo(int[] arr) {
        for (int pos = 0 ; pos < MAX; pos++)
            arr[pos]=pos*2;
    }
}
```


# Ejemplo



```
public class Clase_5_Ejemplo_4 {
    final static int MAX = 5;

    public static void main (String[] args) {
        int[] B=new int[MAX];
        int a = 20;
        cargar_arreglo(B);
        System.out.println("Los datos son:");
        for (int pos = 0 ; pos < MAX; pos++)
            System.out.println(B[pos]);
        cargar_variable_simple(a);
        System.out.println("La variable es:"+a);
    }
    private static void cargar_variable_simple(int c) {
        c = 10;
    }
    private static void cargar_arreglo(int[] arr) {
        for (int pos = 0 ; pos < MAX; pos++)
            arr[pos]=pos*2;
    }
}
```

# Ejemplo



```
public class Clase_5_Ejemplo_4 {
    final static int MAX = 5;

    public static void main (String[] args) {
        int[] B=new int[MAX];
        int a = 20;
        cargar_arreglo(B);
        System.out.println("Los datos son:");
        for (int pos = 0 ; pos < MAX; pos++)
            System.out.println(B[pos]);
        cargar_variable_simple(a);
        System.out.println("La variable es:"+a);
    }
    private static void cargar_variable_simple(int c) {
        c = 10;
    }
    private static void cargar_arreglo(int[] arr) {
        for (int pos = 0 ; pos < MAX; pos++)
            arr[pos]=pos*2;
    }
}
```

# Ejemplo

```
public class Clase_5_Ejemplo_4 {  
    final static int MAX = 5;  
    public static void main (String[] args) {  
        int[] B=new int[MAX];  
        int a = 20;  
        cargar_arreglo(B);  
        System.out.println("Los datos son:");  
        for (int pos = 0 ; pos < MAX; pos++)  
            System.out.println(B[pos]);  
        cargar_variable_simple(a);  
        System.out.println("La variable es:"+a);  
    }  
    private static void cargar_variable_simple(int c) {  
        c = 10;  
    }  
    private static void cargar_arreglo(int[] arr) {  
        for (int pos = 0 ; pos < MAX; pos++)  
            arr[pos]=pos*2;  
    }  
}
```

# Consideraciones en arreglos

Teniendo en cuenta el tipo de pasaje de parámetros en java, devolver un arreglo no estaría bien siempre y cuando esté trabajando sólo sobre el arreglo original.

Por ejemplo:

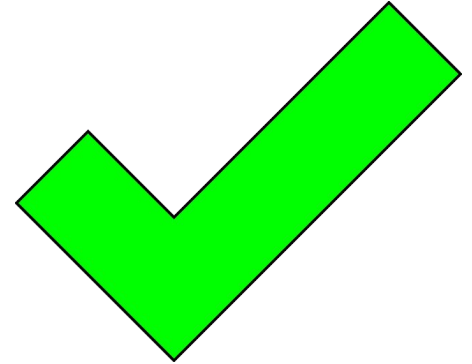
```
public class Clase_5_Ejemplo_5 {  
    final static int MAX = 5;  
    public static void main (String[] args) {  
        int[] A=new int[MAX], B=new int[MAX];  
        B=cargar_arreglo(A);  
    }  
    public static int[] cargar_arreglo(int[] arr) {  
        for (int pos = 0 ; pos < MAX; pos++)  
            arr[pos]=pos*2;  
        }  
        return arr;  
    }  
}
```



# Consideraciones en arreglos

Teniendo en cuenta el tipo de pasaje de parámetros en java, devolver un arreglo no estaría bien siempre y cuando esté trabajando sólo sobre el arreglo original.  
Por ejemplo:

```
public class Clase_5_Ejemplo_5 {  
    final static int MAX = 5;  
    public static void main (String[] args) {  
        int[] A=new int[MAX];  
        cargar_arreglo(A);  
    }  
    public static void cargar_arreglo(int[] arr) {  
        for (int pos = 0 ; pos < MAX; pos++)  
            arr[pos]=pos*2;  
    }  
}
```

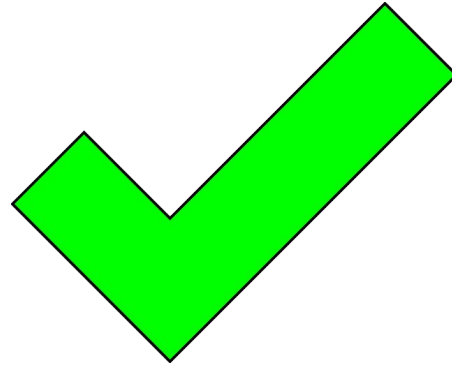


# Consideraciones en arreglos



Pero si quiero devolver un arreglo que es resultado de recorrer y procesar otro, podría ser una buena estrategia para resolver un cierto problema. Por ejemplo:

```
public class Clase_5_Ejemplo_6 {  
    final static int MAX = 5;  
    public static void main (String[] args) {  
        int[] A=new int[MAX], B=new int[MAX];  
        cargar_arreglo(A);  
        B=procesar(A);  
    }  
    public static void cargar_arreglo(int[] arr) {  
        for (int pos = 0 ; pos < MAX; pos++)  
            arr[pos]=pos*2;  
    }  
    public static int[] cargar_arreglo(int[] arr) {  
        int[] resultado=new int[MAX];  
        for (int pos = 0 ; pos < MAX; pos++)  
            resultado=arr[pos]*arr[pos]+5;  
        return resultado;  
    }  
}
```



Para este caso necesito  
conservar los dos: el original  
**A** y el resultado procesado **B**



# ¿Cómo trabajar con arreglos?



Una forma de encarar los ejercicios de arreglos es partir de un esquema en pseudocódigo como el que sigue (**cada uno puede optar por hacerlo o no, los pseudocódigos no se piden en la resolución**):

Definir constantes;

Dentro de main(){

    Definir variables

    Inicializar arreglo/s;

    Cargar arreglo/s;

    Procesar;

    Imprimir arreglos/s;

}

# Corrimiento a derecha

Generalmente asociado al hecho de **insertar** un elemento en un arreglo (ordenado o no)

El usuario quiere insertar un elemento en la quinta posición, entonces:

12	3	6	23	3	8	9	7	4	6	← Valores
0	1	2	3	4	5	6	7	8	9	← Posiciones

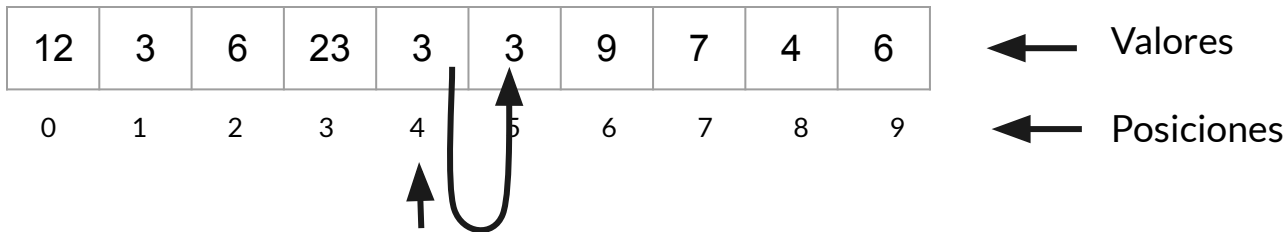
↑

posición 4, que no es la cuarta sino la quinta porque arrancamos de 0

# Corrimiento a derecha

Generalmente asociado al hecho de **insertar** un elemento en un arreglo (ordenado o no)

El usuario quiere insertar un elemento en la quinta posición, entonces:



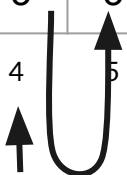
posición 4, que no es la cuarta sino la quinta porque arrancamos de 0

# Corrimiento a derecha

Generalmente asociado al hecho de **insertar** un elemento en un arreglo (ordenado o no)

El usuario quiere insertar un elemento en la quinta posición, entonces:

12	3	6	23	3	3	9	7	4	6
0	1	2	3	4	5	6	7	8	9



← Valores

← Posiciones

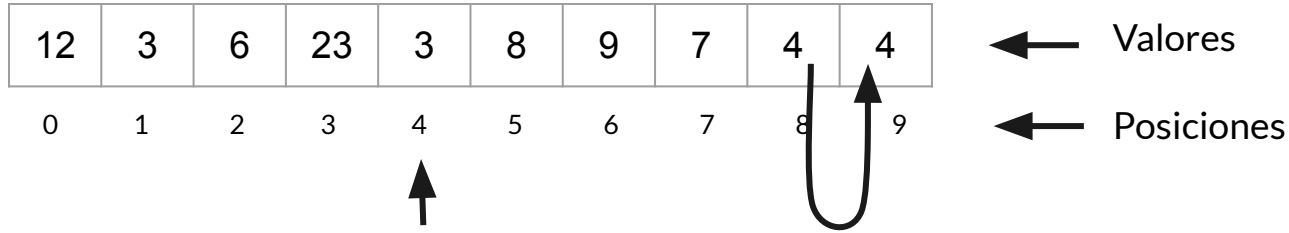
posición 4, que no es la cuarta sino la quinta porque arrancamos de 0



# Corrimiento a derecha

Generalmente asociado al hecho de **insertar** un elemento en un arreglo (ordenado o no)

El usuario quiere insertar un elemento en la quinta posición, entonces:



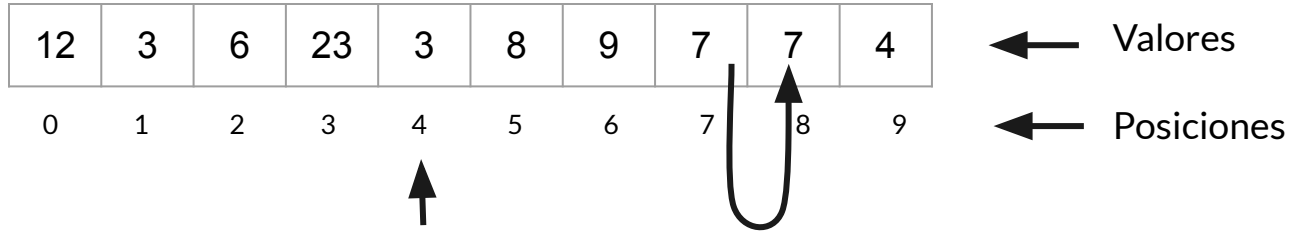
posición 4, que no es la cuarta sino la quinta porque arrancamos de 0

**Importante:** Se deberán desplazar un lugar a derecha todos los elementos que están a la derecha de la cuarta posición.

# Corrimiento a derecha

Generalmente asociado al hecho de **insertar** un elemento en un arreglo (ordenado o no)

El usuario quiere insertar un elemento en la quinta posición, entonces:



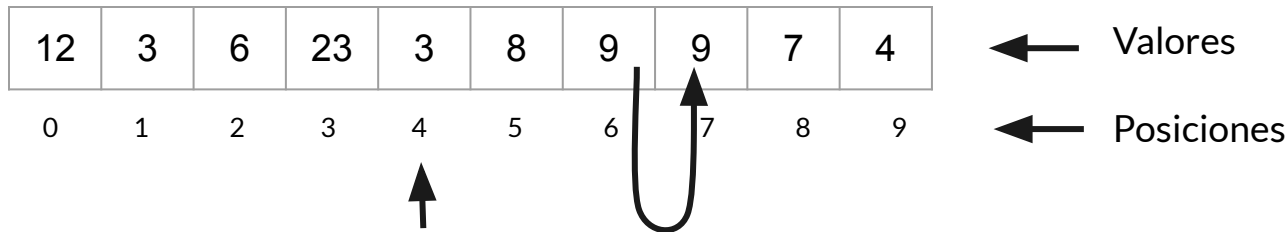
posición 4, que no es la cuarta sino la quinta porque arrancamos de 0

**Importante:** Se deberán desplazar un lugar a derecha todos los elementos que están a la derecha de la cuarta posición.

# Corrimiento a derecha

Generalmente asociado al hecho de **insertar** un elemento en un arreglo (ordenado o no)

El usuario quiere insertar un elemento en la quinta posición, entonces:



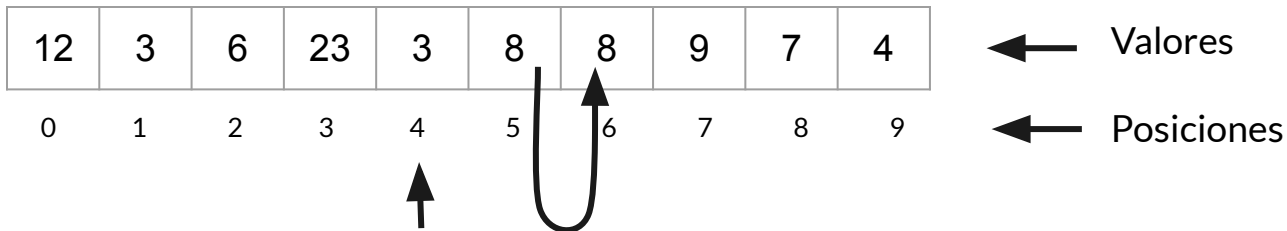
posición 4, que no es la cuarta sino la quinta porque arrancamos de 0

**Importante:** Se deberán desplazar un lugar a derecha todos los elementos que están a la derecha de la cuarta posición.

# Corrimiento a derecha

Generalmente asociado al hecho de **insertar** un elemento en un arreglo (ordenado o no)

El usuario quiere insertar un elemento en la quinta posición, entonces:



posición 4, que no es la cuarta sino la quinta porque arrancamos de 0

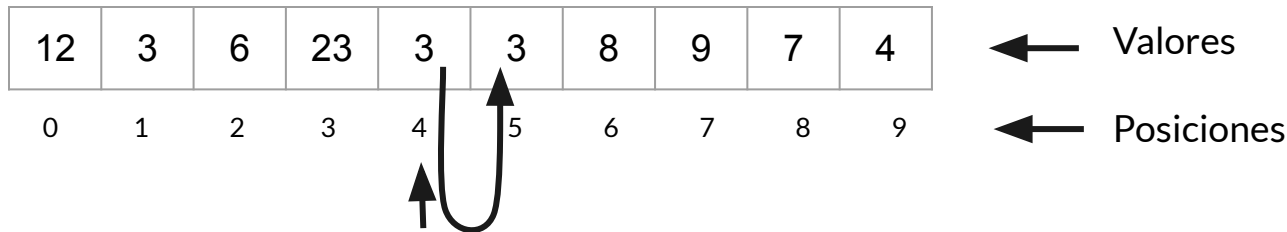
**Importante:** Se deberán desplazar un lugar a derecha todos los elementos que están a la derecha de la cuarta posición.



# Corrimiento a derecha

Generalmente asociado al hecho de **insertar** un elemento en un arreglo (ordenado o no)

El usuario quiere insertar un elemento en la quinta posición, entonces:



posición 4, que no es la cuarta sino la quinta porque arrancamos de 0

- El elemento en la posición `pos`, queda copiado en la posición `pos + 1` y se pierde el último elemento.
- Si tuviera que insertar un elemento en esa posición sería algo como:
  - `arreglo[pos]=valor; // pos es la ingresada por el usuario`

# Corrimiento a izquierda

Generalmente asociado al hecho de **eliminar** un elemento en un arreglo (ordenado o no)

El usuario quiere eliminar el elemento de la séptima posición, entonces:

12	3	6	23	3	8	9	7	4	6
0	1	2	3	4	5	6	7	8	9

← Valores

← Posiciones

↑

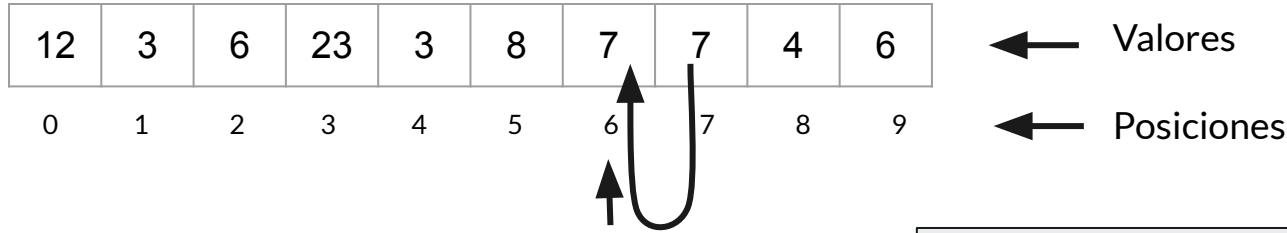
posición 6 porque arrancamos de 0

**Importante:** Se deberán desplazar un lugar a izquierda todos los elementos que están a la derecha de la séptima posición.

# Corrimiento a izquierda

Generalmente asociado al hecho de **eliminar** un elemento en un arreglo (ordenado o no)

El usuario quiere eliminar el elemento de la séptima posición, entonces:



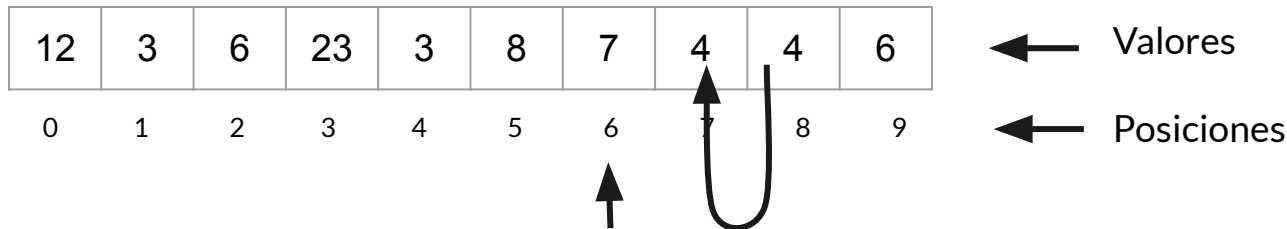
posición 6 porque arrancamos de 0

**Importante:** Se deberán desplazar un lugar a izquierda todos los elementos que están a la derecha de la séptima posición.

# Corrimiento a izquierda

Generalmente asociado al hecho de **eliminar** un elemento en un arreglo (ordenado o no)

El usuario quiere eliminar el elemento de la séptima posición, entonces:



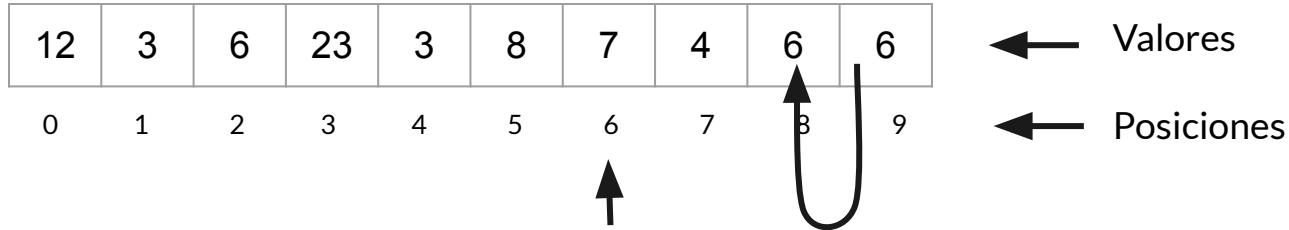
posición 6 porque arrancamos de 0

**Importante:** Se deberán desplazar un lugar a izquierda todos los elementos que están a la derecha de la séptima posición.

# Corrimiento a izquierda

Generalmente asociado al hecho de **eliminar** un elemento en un arreglo (ordenado o no)

El usuario quiere eliminar el elemento de la séptima posición, entonces:



posición 6 porque arrancamos de 0

El último elemento queda duplicado.

# Estructuras ordenadas



Se refiere a una propiedad que posee su composición respecto de sus elementos.

```
private static int buscar_pos_desordenado(int[] arr,int valor) {  
    int pos = 0;  
    while ( (pos<MAX) &&(arr[pos]!=valor)) {  
        pos++;  
    }  
    if (pos<MAX)  
        return pos;  
    else  
        return -1;  
}
```

12	3	6	23	3	8	9	7	4	6
----	---	---	----	---	---	---	---	---	---

```
private static int buscar_pos_ordenado(int[] arr,int valor){  
    int pos = 0;  
    while ( (pos<MAX) &&(arr[pos]>valor)) {  
        pos++;  
    }  
    if ( (pos<MAX) &&(arr[pos]==valor))  
        return pos;  
    else  
        return -1;  
}
```

23	12	9	8	7	6	6	4	3	3
----	----	---	---	---	---	---	---	---	---

# Buscar un elemento en un arreglo ordenado

//Hacer un programa que dado un arreglo de enteros ordenado creciente de tamaño 10, encuentre la posición donde se halla  
//un número entero dado. Si existe, muestre esa posición por pantalla, o indique que no existe.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_5_Ejemplo_7 {
    public static int MAX = 10;
    public static void main (String [] args) {
        int [] arrenteros = new int [MAX];
        cargarArreglo(arrenteros);
        int pos,numero;
        numero=45; // es para el ejemplo pero se puede pedir por consola por ejemplo
        pos = buscar_pos_arreglo_ord_crec(arrenteros,numero);
        if ((pos<MAX)&&(arrenteros[pos]==numero)){
            System.out.println("La posición de "+numero+" es: "+pos);
        }
        else{
            System.out.println(numero + " no existe en el arreglo");
        }
    }
    //La posición que retorna no necesariamente significa que esté ahí, es donde debería estar
    public static int buscar_pos_arreglo_ord_crec(int[] arr,int numero) {
        int pos = 0;
        while ((pos<MAX)&&(arr[pos]<numero)){
            pos++;
        }
        return pos;
    }
}
```



# Métodos de ordenamiento

Los algoritmos de ordenamiento permiten ordenar sus elementos.  
Los métodos más populares son:

- Selección
- Inserción
- Burbujeo



# Selección

```
public static void seleccion(int arr[]) {  
    int i, j, menor, pos, tmp;  
    for (i = 0; i < MAX; i++) { // tomamos como menor el primero  
        menor = arr[i]; // de los elementos que quedan por ordenar  
        pos = i; // y guardamos su posición  
        for (j = i + 1; j < MAX; j++){ // buscamos en el resto  
            if (arr[j] < menor) { // del array algún elemento  
                menor = arr[j]; // menor que el actual  
                pos = j;  
            }  
        }  
        if (pos != i){ // si hay alguno menor se intercambia  
            tmp = arr[i];  
            arr[i] = arr[pos];  
            arr[pos] = tmp;  
        }  
    }  
}
```

# Inserción



```
public static void insercion(int arr[]){  
    for (int i = 1; i < MAX; i++) {  
        int aux = arr[i];  
        int j = i - 1;  
        while ((j >= 0) && (arr[j] > aux)){  
            arr[j+1] = arr[j];  
            j--;  
        }  
        arr[j+1] = aux;  
    }  
}
```

# Burbujeo



```
public static void burbujeo(int[] arr){
    int temp;
    for(int i = 1; i < MAX; i++){
        for (int j = 0 ; j < MAX - 1; j++){
            if (arr[j] > arr[j+1]){
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```

# Arreglos de secuencias



En general se conforman de secuencias de valores con separadores definidos:

Arreglo de enteros con separador= 0

0	3	4	0	8	0	17	20	23	0
---	---	---	---	---	---	----	----	----	---

Arreglo de caracteres con separador = ' '

	h	o	l	a		M	U	N	D	O	
--	---	---	---	---	--	---	---	---	---	---	--

# Arreglos de secuencias



Los arreglos son estructuras muy usadas para la implementación de sistemas y el concepto de secuencias aporta soluciones para problemas específicos de dominios reales. Por Ejemplo:

0	3	4	0	8	0	17	20	23	0
---	---	---	---	---	---	----	----	----	---

Cada secuencia en este arreglo podría hacer referencia a la hora en cual se ha observado algún fenómeno durante las 24 hs cada día. En este caso tenemos que el primer día se observó a las 3 y 4 de la mañana, en el segundo día solo a las 8, en el tercer día a las 17, 20 y 23 horas.

O bien podrían ser valores de algún sensor que tomó datos, en la primera toma pudo obtener 3 y 4 (podría ser una magnitud en algún dominio, por ejemplo temperatura), en la segunda 8 y por último 17, 20 y 23. Esto en una lógica de negocio podría ser que se tomaron datos en tres momentos del día y esos fueron los resultados.

Como estos hay muchísimos más ejemplos de donde se puede aplicar este concepto.

# Arreglos de secuencias



- Trabajar con estas estructuras requiere de métodos que simplifiquen su recorrido, procesar las secuencias, incorporar secuencias, y eliminar secuencias.
- Para ello vamos a partir de una solución para cargar de forma aleatoria un arreglo de secuencias, que nos va permitir utilizarlo para probar nuestros ejercicios.
- Los métodos de carga de secuencias no se van a pedir en la resolución de ejercicios.
- A continuación se dejan ejemplos de códigos (carga e impresión) para arreglos de secuencias de caracteres letras minúsculas separadas por espacios, y de números enteros entre 1 y 9 separados por 0.

# Arreglos de secuencias 1/4



```
import java.util.Random;
public class Clase_5_Ejemplo_8 {
    public static final int MAX = 40, MAXVALOR = 9, MINVALOR = 1;
    public static final double probabilidad_letra = 0.4, double probabilidad_numero = 0.4;
    public static void main(String[] args) {
        char [] arrchar= new char[MAX];
        int [] arrint = new int[MAX];
        cargar_arreglo_aleatorio_secuencias_char(arrchar);
        imprimir_arreglo_secuencias_char(arrchar);
        cargar_arreglo_aleatorio_secuencias_int(arrint);
        imprimir_arreglo_secuencias_int(arrint);
    }
    public static void cargar_arreglo_aleatorio_secuencias_char(char [] arr){
        Random r = new Random();
        arr[0] = ' ';
        arr[MAX-1] = ' ';
        for (int pos = 1; pos < MAX-1; pos++){
            if (r.nextDouble()>probabilidad_letra){
                arr[pos]=(char) (r.nextInt(26) + 'a');
            }
            else{
                arr[pos]=' ';
            }
        }
    }
}
```

# Arreglos de secuencias 2/4

```
public static void imprimir_arreglo_secuencias_char(char [] arr){
    System.out.print("Arreglo de secuencias char\n|");
    for (int pos = 0; pos < MAX; pos++){
        System.out.print(arr[pos]+"|");
    }
    System.out.print("\n");
}

public static void cargar_arreglo_aleatorio_secuencias_int(int [] arr){
    Random r = new Random();
    arr[0] = 0;
    arr[MAX-1] = 0;
    for (int pos = 1; pos < MAX-1; pos++){
        if (r.nextDouble()>probabilidad_numero){
            arr[pos]=(r.nextInt(MAXVALOR-MINVALOR+1) + MINVALOR);
        }
        else{
            arr[pos]=0;
        }
    }
}

public static void imprimir_arreglo_secuencias_int(int [] arr){
    System.out.print("Arreglo de secuencias int\n|");
    for (int pos = 0; pos < MAX; pos++){
        System.out.print(arr[pos]+"|");
    }
    System.out.print("\n");
}
}
```



# Arreglos de secuencias 3/4

//Hacer un programa que dado el arreglo definido y precargado, imprima lo que suma el contenido de cada secuencia.

```
import java.util.Random;
public class Clase_5_Ejemplo_9 {
    public static final int MAX = 20, MAXVALOR = 9, MINVALOR = 1;
    public static final double probabilidad_numero = 0.4;
    public static void main(String[] args) {
        int[] arr = new int[MAX];
        cargar_arreglo_aleatorio_secuencias_int(arr); //REUTILIZAMOS
        imprimir_suma_cada_secuencia(arr);
    }
    public static void imprimir_suma_cada_secuencia(int[] arr){
        int inicio=0,fin=-1,suma;
        while ((inicio < MAX)){
            inicio = obtener_inicio_secuencia(arr,fin+1);
            if (inicio < MAX){
                fin = obtener_fin_secuencia(arr,inicio);
                suma = obtener_suma_secuencia(arr,inicio,fin);
                System.out.println("La suma de la secuencia de "+inicio+" a "+fin+" es "+suma);
            }
        }
    }
}
```

# Arreglos de secuencias 4/4



```
public static int obtener_fin_secuencia(int[] arrEnteros, int ini) {
    while (ini<MAX && arrEnteros[ini]!=0)
        ini++;
    return ini-1;
}

public static int obtener_inicio_secuencia(int[] arrEnteros, int ini) {
    while (ini<MAX && arrEnteros[ini]==0)
        ini++;
    return ini;
}

public static int obtener_suma_secuencia(int[] arr, int inicio, int fin){
    int suma = 0;
    while (inicio <= fin){
        suma+=arr[inicio];
        inicio++;
    }
    return suma;
}
```

# Algunos tips

- Cuando trabajo sobre un arreglo en un método no se debe retornar el mismo que se pasa por parámetro ya que se modifica ese mismo. Si se puede devolver otro arreglo con resultados calculados a partir de ese original.
- El pasaje de parámetros es por copia y cuando trabajo con tipos de referencia lo que se copia es la dirección de memoria no el valor.
- El uso de la propiedad length depende del lenguaje, si no está disponible se debería poner el tamaño en una constante.
- Siempre verificar los límites para acceder a posiciones válidas.
- **Insertar** un elemento en un arreglo implica desplazar a **derecha** en el lugar que debe ir. Si es una secuencia se debe desplazar n veces el tamaño de la secuencia que se quiere insertar. Se pierden los últimos n elementos.
- **Eliminar** un elemento de un arreglo implica desplazar a **izquierda**. Si es una secuencia se debe desplazar tantas veces como el tamaño de la secuencia y quedará duplicado n veces el último elemento.
- Siempre que tengo una estructura ordenada tengo que aprovechar esa característica para cuando quiero trabajar con ella. No es recomendable insertar y luego ordenar porque se desperdician recursos computacionales.
- Las secuencias están separadas por “separadores” que pueden variar dependiendo de la aplicación.