



FUNDACIÓN UNIVERSITARIA DE POPAYÁN

Educación de **calidad** con
responsabilidad **social**



FUNDACIÓN
UNIVERSITARIA
DE POPAYÁN

INGENIERIA DE SISTEMAS

Estructura de Datos

Adrian Danilo Astudillo

Mail: adrian.astudillo@docente.fup.edu.co



FUNDACIÓN
UNIVERSITARIA
DE POPAYÁN

Objetivo

Desarrollar programas informáticos que utilizan estructuras de datos para la solución de problemas propios de la ingeniería, integrando técnicas y herramientas de programación orientada a objetos

CONTENIDO

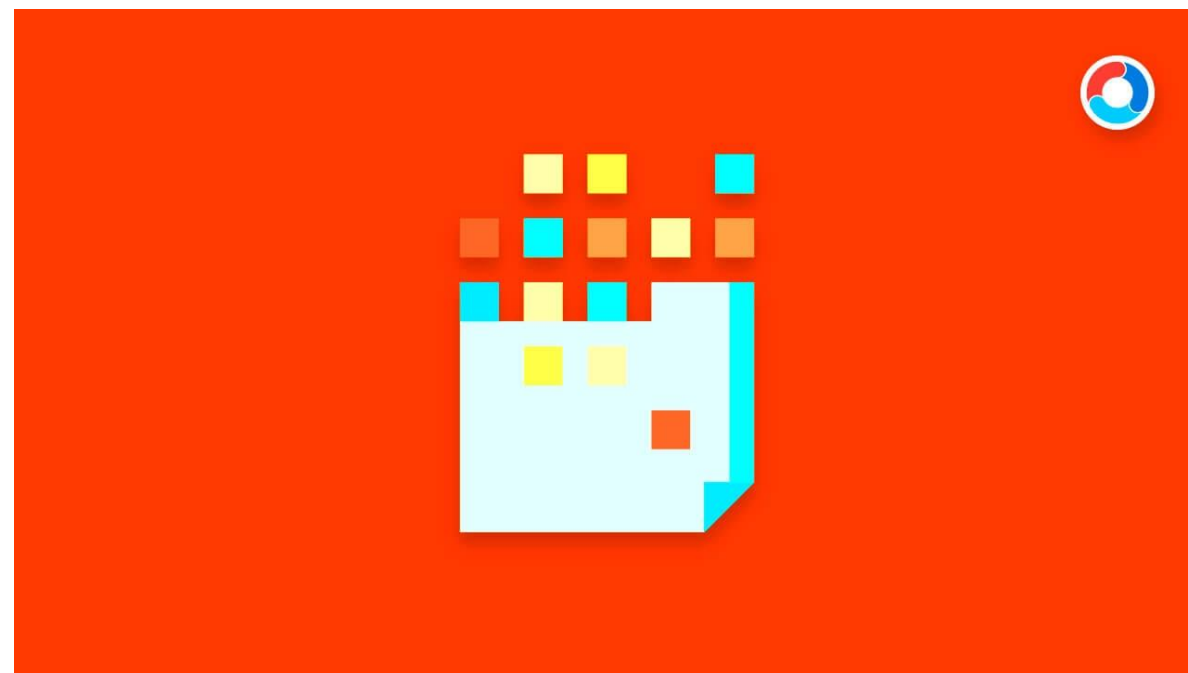
- ✓ Introducción a Estructuras de Datos
- ✓ Sintaxis Array
 - Declaración
 - Definición e inicialización de variables
 - Operadores
 - Estructuras de Control
 - Clases y Objetos



INTRODUCCIÓN

Definición

Las estructuras de datos son formas organizadas y eficientes de almacenar y gestionar datos en programas de computadora. Son esenciales para resolver problemas complejos y optimizar la manipulación de información.



CONTENIDO

- ✓ Introducción a Estructuras de Datos
- ✓ Sintaxis Array
 - Declaración y Inicialización
 - Operaciones con Arreglos
 - Búsqueda de elementos en un arreglo



Java - Declaración

```
//Declaración e instanciación del arreglo en una linea  
tipo nombreArreglo[] = new tipo[MAX];  
O  
tipo nombreArreglo[];  
nombreArreglo[] = new tipo[MAX];  
//lo mismo en dos lineas
```

- **tipo**: es algún tipo de dato válido en java.
- **nombreArreglo** es el nombre del arreglo. Sugerencia que se representativo de los datos que almacena.
- **MAX** es el total de elementos o Casillas que tendrá el arreglo
- Los **[]** pueden ir a la izquierda o a la derecha de **nombreArreglo**

Java – Ejemplo Arreglos

```
//Se declara una constante
final int MAX = 10;
// Arreglo puede almacenar máximo 10 números enteros

int edades[]= new int[MAX];

// Arreglo puede almacenar máximo 10 números de doble precisión

double[] numeros= new double[MAX];

// Arreglo puede almacenar máximo 10 cadenas de caracteres

String[] nombres;
nombres = new String[MAX];
```


Java - Declaración

```
//Pueden Declararse e inicializarse en una sola linea
String nombres[] = {"Julián", "Marcos", "Julián", "Marcos"};

int enteros[]={1,21,18,14};
```

```
//Se conoce del arreglo el máximo de elementos que es cinco
String nombres[] = new String[5];

nombres[0]="Marcos";
nombres[1]="Paola";
nombres[2]="Daniela";
nombres[3]="Germán";
nombres[4]="Oscar";
//Imprime el total de casilla reservadas para el arreglo nombres
System.out.println("Total Casillas para nombres:" +nombres.length);
```

CONTENIDO

- ✓ Introducción a Estructuras de Datos
- ✓ Sintaxis Array
 - Declaración y Inicialización
 - Operaciones con Arreglos
 - Búsqueda de elementos en un arreglo



Java – Operaciones con Arreglo

- La lectura y la impresión en java deben realizarse elemento por elemento.
- No es posible leer e imprimir todos los elementos en un sola instrucción.

```
// Lectura o llenado del arreglo
for (i=0; i<totalEdades; i++) {
    System.out.println("\nIngrese edad:");
    edades[i]=lee.nextInt();
}
```

```
// Impresión del arreglo
System.out.println("\nEdades:");
for (i=0; i<totalEdades; i++) {
    System.out.println(edades[i]+ " ");
}
```

Java – Ejemplo

```
package ed;
import java.util.Scanner; // Libreria Java que toma Datos de Entrada por teclado
public class ED {
    /**
     * Método Principal punto donde inicia el programa
     */
    public static void main(String[] args) {
        Scanner lec = new Scanner(source: System.in); // Declaración de Variable que toma dato por el teclado
        final int totalEdades = 10; // Se declara una Constante
        int edades[] = new int[totalEdades]; // Arreglo puede almacenar máximo 10 números enteros
        // Ciclo que hace un recorrido de Diez repeticiones para agregar elementos del arreglo
        for(int i=0;i<totalEdades;i++){
            System.out.println(x: "\n Ingrese Edad: "); //Mensaje de texto en pantalla
            edades[i]=lec.nextInt(radix: totalEdades); //Guardamos el elemento en el arreglo
        }
        // Ciclo que hace un recorrido de Diez repeticiones para imprimir elementos del arreglo
        System.out.println(x: "\n Impresión de Edades: "); //Mensaje de texto en pantalla
        for(int i=0;i<totalEdades;i++){
            System.out.println(edades[i]+ " "); //Imprime el elemento del arreglo
        }
    }
}
```

CONTENIDO

- ✓ Introducción a Estructuras de Datos
- ✓ Sintaxis Array
 - Declaración y Inicialización
 - Operaciones con Arreglos
 - **Búsqueda de elementos en un arreglo**
 - Ordenar los elementos (Método Burbuja)



Java – Buscar un Elemento de un Arreglo

```
// Buscar un elemento en el arreglo
System.out.print(s: "Ingrese el valor a buscar: ");
int elementoBuscado = lec.nextInt();
boolean encontrado = false;

for (int i = 0; i < 3; i++) {
    if (edades[i] == elementoBuscado) {
        System.out.println("El valor " + elementoBuscado + " se encuentra en la posición " + i + "
        encontrado = true;
        break;
    }
}

if (!encontrado) {
    System.out.println("El valor " + elementoBuscado + " no se encuentra en el arreglo.");
}

- }
```


CONTENIDO

- ✓ Introducción a Estructuras de Datos
- ✓ Sintaxis Array
 - Declaración y Inicialización
 - Operaciones con Arreglos
 - Búsqueda de elementos en un arreglo
 - Ordenar los elementos (Ordenamiento Burbuja)



Ordenamiento Burbuja

Ahora que puedes hacer malabarismos con los elementos de los arreglos, es hora de aprender como ordenarlos. Se han inventado muchos algoritmos de clasificación, que difieren mucho en velocidad, así como en complejidad. Vamos a mostrar un algoritmo muy simple, fácil de entender, pero desafortunadamente, tampoco es muy eficiente. Se usa muy raramente, y ciertamente no para listas extensas.

Digamos que una lista se puede ordenar de dos maneras:

Ascendente (o más precisamente, no descendente): si en cada par de elementos adyacentes, el primer elemento no es mayor que el segundo.

Descendente (o más precisamente, no ascendente): si en cada par de elementos adyacentes, el primer elemento no es menor que el segundo.

En las siguientes secciones, ordenaremos la lista en orden ascendente, de modo que los números se ordenen de menor a mayor.

Aquí está la lista:

8	10	6	2	4
---	----	---	---	---

Ordenamiento Burbuja

Intentaremos utilizar el siguiente enfoque: tomaremos el primer y el segundo elemento y los compararemos; si determinamos que están en el orden incorrecto (es decir, el primero es mayor que el segundo), los intercambiaremos; Si su orden es válido, no haremos nada. Un vistazo a nuestra arreglo confirma lo último: los elementos 01 y 02 están en el orden correcto, así como $8 < 10$.

8	10	6	2	4
---	----	---	---	---

Ahora observa el segundo y el tercer elemento. Están en las posiciones equivocadas. Tenemos que intercambiarlos:

8	6	10	2	4
---	---	----	---	---

Ordenamiento Burbuja

Vamos más allá y observemos los elementos tercero y cuarto. Una vez más, esto no es lo que se supone que es. Tenemos que intercambiarlos:

8	6	2	10	4
---	---	---	----	---

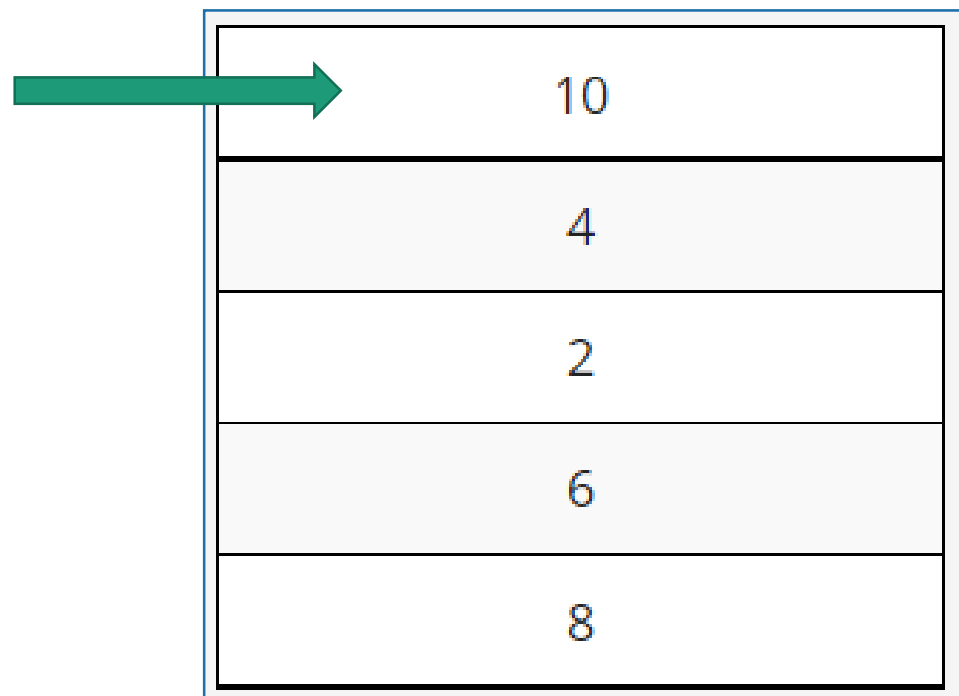
Ahora comprobemos los elementos cuarto y quinto. Si, ellos también están en las posiciones equivocadas. Ocurre otro intercambio:

8	6	2	4	10
---	---	---	---	----

Ordenamiento Burbuja

El primer paso a través del arreglo ya está terminado. Todavía estamos lejos de terminar nuestro trabajo, pero algo curioso ha sucedido mientras tanto. El elemento más grande, 10, ya ha llegado al final de la lista. Ten en cuenta que este es el lugar deseado para él. Todos los elementos restantes forman un lío pintoresco, pero este ya está en su lugar.

Ahora, por un momento, intenta imaginar arreglo de una manera ligeramente diferente, es decir, de esta manera:



10
4
2
6
8

Observa - El 10 está en la parte superior. Podríamos decir que flotó desde el fondo hasta la superficie, al igual que las burbujas en una copa de champán. El método de clasificación deriva su nombre de la misma observación: se denomina ordenamiento de burbuja.

Ordenamiento Burbuja

Ahora comenzamos con el segundo paso a través del arreglo. Miramos el primer y el segundo elemento, es necesario un intercambio:

6	8	2	4	10
---	---	---	---	----

Tiempo para el segundo y tercer elemento: también tenemos que intercambiarlos:

6	2	8	4	10
---	---	---	---	----

Ahora el tercer y cuarto elementos, y la segunda pasada, se completa, ya que 8 ya está en su lugar:

6	2	4	8	10
---	---	---	---	----

Ordenamiento Burbuja

Comenzamos el siguiente pase inmediatamente. Observe atentamente el primer y el segundo elemento: se necesita otro cambio:

2	6	4	8	10
---	---	---	---	----

Ahora 6 necesita ir a su lugar. Cambiamos el segundo y el tercer elemento:

2	4	6	8	10
---	---	---	---	----

La lista ya está ordenada. No tenemos nada más que hacer. Esto es exactamente lo que queremos.

Como puedes ver, la esencia de este algoritmo es simple: comparamos los elementos adyacentes y, al intercambiar algunos de ellos, logramos nuestro objetivo.

Ordenamiento Burbuja – Código Java

```
// Ordenamiento Burbuja Java
public class BubbleSort {
    public static void main(String[] args) {
        int[] numeros = {8, 10, 6, 2, 4};

        for (int i = 0; i < numeros.length - 1; i++) {
            for (int j = 0; j < numeros.length - i - 1; j++) {
                if (numeros[j] > numeros[j + 1]) {
                    // Intercambio de elementos
                    int temp = numeros[j];
                    numeros[j] = numeros[j + 1];
                    numeros[j + 1] = temp;
                }
            }
        }
        System.out.println("Arreglo ordenado:");
        for (int numero : numeros) {
            System.out.print(numero + " ");
        }
    }
}
```

Ordenamiento Burbuja – Explicación

El bucle externo (índice i) controla cuántas veces se repite el proceso de comparación e intercambio. El bucle interno (índice j) recorre el arreglo y compara pares de elementos adyacentes. Si el elemento en la posición j es mayor que el elemento en la posición $j + 1$, se realiza un intercambio.

Con cada iteración del bucle externo, el elemento más grande "flota" hacia la última posición. Después de completar todas las iteraciones, el arreglo estará ordenado de manera ascendente.

Recuerda que, aunque el ordenamiento de burbuja es útil para aprender sobre algoritmos de ordenamiento, en aplicaciones del mundo real, a menudo se prefieren algoritmos más eficientes como el ordenamiento rápido o el ordenamiento de mezcla para conjuntos de datos más grandes.



FUNDACIÓN
UNIVERSITARIA
DE POPAYÁN

Gracias