


# Tema 11


## Organización lógica de los datos

### Estructuras estáticas

 Fundamentos esenciales de la programación estructurada, orientada a objetos y funcional.

# 1. Introducción

La **organización lógica de los datos** define cómo se estructuran y manipulan desde una perspectiva abstracta, **independiente del hardware**.

 Es la base para:

- Diseño eficiente de algoritmos
- Reutilización de código
- Claridad y escalabilidad en programas

## 2. Organización lógica vs. física

Aspecto	Organización lógica	Organización física
Nivel	Conceptual	Concreto (memoria/disco)
Estructura	Abstracta	Disposición real
Impacto	Diseño de código	Rendimiento físico

- ✓ Coherencia
- ⚡ Eficiencia
- ⚙ Modularidad
- ↺ Reutilización

### 3. Tipos de datos escalares

- **Enteros:** complemento a 2
- **Reales:** IEEE 754
- **Carácter:** Unicode (UTF-8, UTF-16)
- **Booleano:** 0 / 1
- **Enumerado:** conjunto finito
- **Rango:** subconjunto acotado

 Forman la base para estructuras más complejas

## 4. Tipos de datos estructurados

### Vectores (arrays)

- Memoria contigua
- Acceso directo por índice
- Unidimensional o multidimensional

### Conjuntos

- Elementos únicos
- Operaciones: unión  $\cup$ , intersección  $\cap$ , diferencia  $-$

### Registros / tuplas

- Agrupan datos **heterogéneos**
- Ejemplos: `struct`, `record`

## 5. Tipos abstractos de datos (TAD)

Los TAD definen:

- Un dominio de valores
- Un conjunto de operaciones válidas
- Propiedades semánticas

## Principales TAD

TAD	Operaciones clave	Estructura
Pila	push , pop (LIFO)	Array o lista
Cola	enqueue , dequeue (FIFO)	Array circular
Lista	Inserciones, recorridos	Array o nodos
Árbol	Insertar, buscar, recorrer	Nodos jerárquicos
Grafo	Adyacencias, caminos	Matriz o lista
Tabla hash	put , get , colisiones	Array indexado

## 6. Implementación estática de estructuras

Estructuras **definidas en arrays** o espacios fijos de memoria.

- ✓ Rápidas
- ✓ Predecibles
- ✓ Usadas en entornos controlados



## 6.1 Pilas

- Implementación: array + tope ( `int top` )
- Operaciones: `push()` , `pop()`
- Usos:
  - Llamadas
  - Backtracking
  - Evaluación de expresiones

## 6.2 Colas y deque

- **Cola:** array circular
  - Dos punteros: inicio y fin
- **Deque:** inserción/extracción en ambos extremos

 Usos: buffers, planificación, tiempo real

## 6.3 Listas (estáticas)

- Array con desplazamientos
- Inserciones/eliminaciones lentas ( $O(n)$ )
- Acceso aleatorio  $O(1)$

 Pueden simular estructuras dinámicas en entornos de memoria limitada

## 6.4 Árboles (estáticos)

### Árbol binario

- Nodo con hasta 2 hijos
- Recorridos: preorden, inorden, postorden

### Árbol de búsqueda (BST)

- Orden: izquierda < nodo < derecha
- Eficiencia si está equilibrado

## Árboles balanceados

- AVL / Rojo-Negro: equilibrio automático
- Mejora inserciones, búsquedas

## Árboles n-arios

- Más de 2 hijos por nodo
- Usos: DOM, expresiones, estructuras jerárquicas

## Heap (montículo)

- Árbol binario completo

Tipo	Propiedad
Max-heap	Nodo $\geq$ hijos
Min-heap	Nodo $\leq$ hijos

 Usos: colas de prioridad, ordenaciones

## 6.5 Grafos

### Representaciones

Tipo	Características
Matriz de adyacencia	Rápida, costosa en memoria
Lista de adyacencia	Eficiente en grafos dispersos

 Usos: algoritmos de rutas, redes, grafos sociales

## 6.6 Tablas hash

- Array indexado por función hash
- Acceso promedio  $O(1)$

### Colisiones

- Encadenamiento (listas)
- Direccionamiento abierto (linear probing, etc.)

🧩 Usos: diccionarios, caches, autenticación



## 6.7 Diccionarios

- Colección de **pares clave/valor**
- Implementados con:
  - `HashMap` en Java
  - `unordered_map` en C++

 Usos: almacenamiento, configuración, datos etiquetados

## 7. Utilidad en programación y competencias

 Las estructuras **estáticas** son fundamentales en:

- Entornos con restricciones (embedded, concursos)
- Entrevistas técnicas y pruebas tipo OI, ProgramaMe
- Algoritmos con necesidad de rendimiento garantizado

 Rápidas, seguras y previsibles

## Conclusión

- ✓ La organización lógica permite estructurar datos de forma eficiente
- ✓ Las estructuras estáticas ofrecen velocidad y simplicidad
- ✓ Entender sus ventajas es clave en desarrollo y optimización
- 💡 Dominar estas estructuras es esencial para el programador moderno