

# Tema 12

## Organización lógica de los datos

### Estructuras dinámicas

 Fundamentos de estructuras flexibles y escalables mediante memoria dinámica.


## ◆ 1. Introducción

Las estructuras dinámicas permiten:

- Almacenar y modificar datos sin tamaño fijo
- Gestionarse en **tiempo de ejecución**
- Usar **punteros o referencias** para enlazar nodos

🎯 Clave para software **flexible, escalable y eficiente**

## 2. Organización lógica de los datos

 Modelo abstracto para:

- Relacionar y manipular datos
- Independiente del almacenamiento físico
- Base para algoritmos y estructuras eficientes

## Lógica vs. Física

Organización lógica	Organización física
Conceptual	Concreta (memoria/disco)
Estructura abstracta	Disposición real
Afecta al código fuente	Afecta al rendimiento

## **Beneficios de la organización lógica**

- ✓ **Coherencia:** estructuras claras y predecibles
- ⚡ **Eficiencia:** mejor uso de recursos
- ⚙️ **Modularidad:** programas escalables
- 🔄 **Reutilización:** estructuras aplicables a distintos problemas
- 📖 Base de programación estructurada, orientada a objetos y funcional



### 3. Fundamentos técnicos

- Asignación dinámica de memoria: nodos creados y eliminados en tiempo de ejecución
- Enlaces por **punteros o referencias**
- Memoria **no contigua**
- Inserción y eliminación eficientes (no requieren desplazamiento)
- Acceso más lento que arrays (sin índice directo)
- Mayor complejidad:
  - Manual (C/C++)
  - Automática (Java, Python)

#### ⚠ Riesgos:

- Fugas de memoria

## **12** 4. Tipos de datos escalares

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

## 5. Tipos de datos estructurados

### 5.1 Vectores (arrays)

- Acceso directo por índice
- Unidimensional o multidimensional

### 5.2 Conjuntos

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

### 5.3 Registros / Tuplas

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



## 6. Tipos Abstractos de Datos (TAD)

Definen:

- Dominio de valores
- Operaciones válidas
- Propiedades semánticas

TAD	Operaciones clave
Pila	push , pop (LIFO)
Cola	enqueue , dequeue (FIFO)
Lista	insertar , eliminar
Árbol	insertar , buscar , recorrer
Grafo	Relaciones arbitrarias

## 7. Listas dinámicas

## 7.1 Listas simples

- Nodo: dato + puntero al siguiente
- Operaciones: insertar, eliminar, buscar, recorrer

## 7.2 Listas doblemente enlazadas

- Nodo: puntero anterior y siguiente
- Navegación en ambas direcciones
- Mejora eficiencia de eliminación y recorrido

## 7.3 Listas circulares

- Último nodo apunta al primero
- Útiles en buffers, bucles y estructuras cíclicas

## 8. Pilas y colas dinámicas

## 8.1 Pilas (LIFO)

- `push()` , `pop()` , `top()`
- Usos:
  - Backtracking
  - Gestión de llamadas
  - Evaluación de expresiones

## 8.2 Colas (FIFO)

- `enqueue()` , `dequeue()` , `front()`
- Usos:
  - Planificación de procesos
  - Comunicación entre sistemas
  - Simulación



## 8.3 Deques

- Inserción y extracción por **ambos extremos**
- Más flexibles que pilas o colas
- Usos: algoritmos de ventana deslizando, buffers

## 9. Árboles dinámicos

## Árbol binario

- Cada nodo con **hasta 2 hijos**
- Recorridos: preorden, inorden, postorden

## Árboles de búsqueda (BST)

- Orden natural: `izq < nodo < der`
- Búsqueda eficiente si equilibrado

## Árboles balanceados

- **AVL:** rebalanceo por rotaciones
- **Rojo-Negro:** balanceo más relajado, ideal en inserciones frecuentes

 Mantienen rendimiento óptimo


## Árboles n-arios

- Nodos con más de dos hijos
- Usos: DOM, menús, árboles de sintaxis

## Heap (Montículo)

- Árbol binario completo

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

 Usos: colas de prioridad, ordenaciones (Heapsort)

## 10. Grafos dinámicos

### Representación común

- Lista de adyacencia
  - Cada nodo tiene una lista de vecinos
  - Estructura eficiente para grafos dispersos

 Nodos y aristas pueden ser objetos con referencias

 Aplicaciones:

- Rutas, navegación
- Redes sociales
- IA: Dijkstra,  $A^*$ , etc.



## 11. Tablas hash con encadenamiento

- Estructura: array + función hash
- Colisiones: listas enlazadas

### Características

- Acceso promedio  $O(1)$
- Resolución de colisiones: encadenamiento externo

### Usos:

- Caches
- Diccionarios
- Bases de datos (índices hash)

## 12. Aplicaciones reales

### Software moderno:

- Juegos e IA: nodos de comportamiento
- Bases de datos: árboles B y B+ para índices
- Compiladores: grafos de flujo, árboles sintácticos
- Navegadores: árbol DOM
- Sistemas operativos: planificación, buffers, colas

## Conclusión

✓ Las estructuras dinámicas permiten:

- Modelar datos sin límites fijos
- Gestionar memoria de forma eficiente
- Crear programas escalables y adaptativos

 Son esenciales en programación avanzada y desarrollo profesional