



INSTITUTO TECNOLÓGICO DE FRONTERA COMALAPA

MATERIA

Estructura de Datos

TEMA 1

Act 1.2 Cuadro comparativo

ALUMNO

Amilcar Alejandro Corona Escobar

Tercer semestre, ing. Sistemas Computacionales.

NC: 241260074

MODALIDAD

Escolarizada

DOCENTE

Ing. Mingo Velázquez Francisco Javier.

Frontera Comalapa, Chiapas, a 01 de septiembre de 2025

Introducción

Las estructuras de datos son como las “cajas” donde guardamos la información dentro de un programa. Dependiendo de cómo organicemos esos datos, será más fácil o más difícil trabajar con ellos. Por eso es importante aprender no solo qué son, sino también cómo se clasifican, cómo se usan en la memoria y qué tan eficientes son los algoritmos que las manejan. Conocer esto nos ayuda a entender mejor cómo funcionan los programas que usamos todos los días y a crear soluciones más rápidas y ordenadas.

Justificación

En la vida real, cuando queremos resolver un problema, no basta con que la solución funcione: también debe ser práctica y eficiente. En programación pasa lo mismo. Si no usamos bien las estructuras de datos o los algoritmos, un programa puede volverse lento, gastar demasiada memoria o incluso fallar. Estudiar este tema nos da las herramientas para organizar la información de forma correcta y para elegir los algoritmos que mejor se adapten a cada situación. En pocas palabras, nos prepara para programar de manera más profesional.

| Tema / Punto | Definición / Concepto | Ejemplo / Aplicación |
| --- | --- | --- |
| Clasificación de Estructuras de Datos | Forma en que se organizan los datos según su relación y acceso. | Lineales: listas, pilas, colas; No lineales: árboles, grafos |
| Tipos de Datos Abstractos (TDA) | Modelos que definen operaciones y comportamiento de los datos sin importar la implementación. | Pila, Cola, Lista, Árbol |
| Ejemplos de TDA’s | Representaciones concretas de los TDA en la programación. | Pila → push/pop; Cola → enqueue/dequeue |
| Manejo de Memoria | Forma en que se reserva y utiliza la memoria para almacenar datos. | Asignación de memoria para un array (estática) o lista enlazada (dinámica) |
| Memoria Estática | Memoria asignada al declarar la estructura, tamaño fijo durante la ejecución. | Array (arreglo) |
| Memoria Dinámica | Memoria asignada y liberada durante la ejecución según necesidad. | Listas enlazadas |
| Análisis de Algoritmos | Evaluación del rendimiento y recursos que utiliza un algoritmo. | Comparar dos algoritmos de ordenamiento: Bubble Sort (O(n²)) vs Merge Sort (O(n log n)) |
| Complejidad en el Tiempo | Mide la cantidad de operaciones que realiza un algoritmo según el tamaño del problema. | Ordenamiento: O(n²) vs O(n log n) |
| Complejidad en el Espacio | Mide la cantidad de memoria que requiere un algoritmo. | Algoritmo recursivo vs iterativo |
| Eficiencia de los Algoritmos | Determina qué tan óptimo es un algoritmo considerando tiempo y espacio. | Elegir merge sort en vez de bubble sort para listas grandes |

Conclusión

En resumen, aprender sobre estructuras de datos no es solo teoría: tiene un impacto directo en la manera en que resolvemos problemas con un programa. Saber elegir entre una lista, una cola o un árbol, entender cómo usar la memoria de forma correcta y analizar qué tan rápido y eficiente es un algoritmo, nos ayuda a ser mejores programadores. Al final, este conocimiento nos permite diseñar aplicaciones que funcionen de manera más rápida, segura y confiable.

Fuentes consultadas

* GeeksforGeeks. (s.f.). *Data Structures - Basics*. Recuperado de: https://www.geeksforgeeks.org/data-structures
* Tutorialspoint. (s.f.). *Data Structures Tutorial*. Recuperado de: https://www.tutorialspoint.com/data\_structures\_algorithms
* Programiz. (s.f.). *Data Structures*. Recuperado de: https://www.programiz.com/dsa