|  |  |  |
| --- | --- | --- |
| Logotipo, nombre de la empresa  El contenido generado por IA puede ser incorrecto. | Imagen que contiene Logotipo  El contenido generado por IA puede ser incorrecto. | Diagrama  El contenido generado por IA puede ser incorrecto. |

**INSTITUTO TECNOLÓGICO DE FRONTERA COMALAPA**

**MATERIA**

**ESTRUCTURA DE DATOS**

**TEMA**

**RESUMEN**

**ESTUDIANTE**

**ISAIAS SALATHIEL LOPEZ TORRES**

**TERCER SEMESTRE, ING. SISTEMAS COMPUTACIONALES.**

**NC:241260085**

**MODALIDAD**

**ESCOLARIZADA**

**DOCENTE**

**ING. FRANCISCO JAVIER MINGO VELAZQUEZ**

**FRONTERA COMALAPA CHIAPAS, A 26 DE AGOSTO DEL 2025**

**INTRODUCCIÓN A LAS ESTRUCTURAS DE DATOS**

1.1 Clasificación de las estructuras de datos

* Las estructuras de datos son formas de organizar la información para que pueda ser utilizada de manera eficiente en un programa.
* Lineales: los datos se organizan en secuencia, uno tras otro. Ejemplos: arreglos, listas, pilas y colas. Su ventaja es la simplicidad en el acceso, pero suelen tener limitaciones en inserciones o eliminaciones.
* No lineales: los datos no siguen una secuencia, sino que se estructuran en jerarquías o conexiones múltiples. Ejemplos: árboles y grafos. Son útiles cuando las relaciones entre los datos son complejas.

1.2 Tipos de datos abstractos (TDA)

* Son modelos matemáticos que definen un conjunto de operaciones sobre los datos sin especificar la forma en que se implementan en la memoria.
* Permiten separar la lógica de uso de la estructura de su implementación, lo que facilita reutilizar y entender mejor los programas.

1.3 Ejemplos de TDA’s

* Pila (Stack): sigue el principio LIFO (Last In, First Out). Ejemplo: el último libro colocado en una pila es el primero que se retira.
* Cola (Queue): sigue el principio FIFO (First In, First Out). Ejemplo: una fila en el banco donde el primero en llegar es el primero en ser atendido.
* Lista enlazada: cada elemento (nodo) tiene un valor y una referencia al siguiente nodo. Permite insertar y eliminar elementos fácilmente, aunque acceder a ellos puede ser más lento que en un arreglo.

1.4 Manejo de memoria

* El uso de memoria es fundamental para que los programas aprovechen eficientemente los recursos del sistema.
* 1.4.1 Memoria estática: se asigna en tiempo de compilación y su tamaño no cambia durante la ejecución. Ejemplo: un arreglo con un tamaño fijo declarado al inicio del programa.
* 1.4.2 Memoria dinámica: se asigna en tiempo de ejecución, lo que permite que la estructura crezca o disminuya según las necesidades. Ejemplo: listas enlazadas o árboles, donde los nodos se crean y eliminan mientras corre el programa.

1.5 Análisis de algoritmos

* Es el proceso de evaluar el rendimiento de un algoritmo en cuanto a tiempo y memoria.
* 1.5.1 Complejidad en el tiempo: mide cuántas operaciones realiza un algoritmo dependiendo del tamaño de la entrada. Se expresa comúnmente con notación Big O (O(n), O(log n), etc.).
* 1.5.2 Complejidad en el espacio: mide la cantidad de memoria necesaria para ejecutar el algoritmo, incluyendo variables, estructuras auxiliares y datos de entrada.
* 1.5.3 Eficiencia de los algoritmos: busca un balance entre rapidez de ejecución y bajo consumo de memoria. Un algoritmo eficiente resuelve el problema correctamente utilizando los menores recursos posibles.

**FUENTES DE INFORMACION**

1. *GeeksforGeeks* — artículos como *“Data Structure Types, Classifications and Applications”* [GeeksforGeeks](https://www.geeksforgeeks.org/dsa/what-is-data-structure-types-classifications-and-applications/?utm_source=chatgpt.com)
2. *GeeksforGeeks* — artículo *“Static Data Structure vs Dynamic Data Structure”* [GeeksforGeeks](https://www.geeksforgeeks.org/dsa/static-data-structure-vs-dynamic-data-structure/?utm_source=chatgpt.com)
3. *Simplilearn* — “What is Data Structure: Types, & Applications” [Simplilearn.com](https://www.simplilearn.com/tutorials/data-structure-tutorial/what-is-data-structure?utm_source=chatgpt.com)
4. *Wikipedia* — “Data structure”