**Dato**

Representaciones numéricas o algorítmicas de un determinado atributo o variable cualitativa o cuantitativa. Esto quiere decir que los datos son información que recibe el computador y que es manipulado mediante el procesamiento de los algoritmos de programación.

**Dato → Información**

Es la expresión de las características puntuales de las entidades sobre las cuales operan dichos algoritmos. Es decir, son el input inicial, a partir del cual puede procesarse y componerse la información.

**Estructura de datos**

Los datos son particularmente importantes para estructura de datos, rama de la computación que estudia la forma particular de almacenaje de la información en porciones mínimas para lograr una posterior recuperación eficiente.

Es una forma particular de organizar datos en una computadora para que puedan ser utilizados de manera eficiente. Diferentes tipos de estructuras de datos son adecuadas para diferentes tipos de aplicaciones, y algunos son altamente especializados para tareas específicas.

**Tipos de datos**

* Primitivos: Tipos de datos simples que pueden tener las variables, es decir, todos los tipos de valores básicos que podemos usar, sin entrar en estructuras más complejas como podrían ser los objetos o arrays.
  + Int (numeros enteros)
  + Double (números decimales más largos)
  + Float (números decimales 32 bits más cortos)
  + Char (un solo carácter)
  + String (cadenas)
  + Boolean (que solo puede tener true or false)
  + Entre otros…
* Abstractos: Es un conjunto de datos u objetos creados de manera personalizada por un programador para un fin específico. Un **TDA** es una abstracción que permite modelar las características de un elemento en particular.
  + Un TDA tendrá una parte que será invisible al usuario la cual hay que proteger y que se puede decir que es irrelevante para el uso del usuario y está constituida tanto por la maquinaria algorítmica que implemente la semántica de las operaciones como por los datos que sirvan de enlace entre los elementos del TDA, es decir, información interna necesaria para la implementación que se está haciendo para ese comportamiento del TDA.

(Resumiendo podemos decir que tanto la implementación de las operaciones como los elementos internos del TDA serán privados al acceso externo ocultos a cualquier otro nivel).

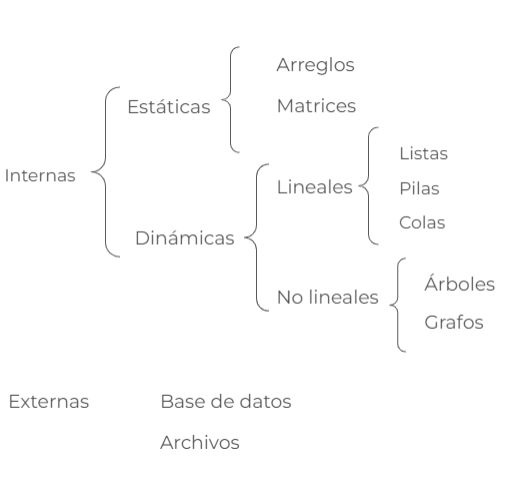
**¿Qué es una estructura de datos?**

Es una forma particular de organizar datos en una computadora para que puedan ser utilizados de manera eficiente. Diferentes tipos de estructuras de datos son adecuados para diferentes tipos de aplicaciones, y algunos son altamente especializados para tareas específicas.

(En pocas palabras son formas de organizar información para manipular, buscar e insertar datos de manera eficiente)

**Tipos de estructura de datos**

* Internas
  + Estáticas
    - Arreglos: Pueden almacenar múltiples datos y pueden ser de diferente tipo de dato. (almacenan múltiples datos en una sola variable)
    - Matrices
  + Dinámicas
    - Lineales
      * Listas: Se van enlazando entre sí. (Se componen de nodos que tienen dos atributos {dato y enlace})
      * Pilas: Se van apilando una después de otras. (solamente pueden eliminar o insertar en la cima de la lista)
      * Colas: Permite simular el comportamiento de una fila o cola de la vida real.
    - No lineales
      * Árboles
      * Grafos
* Externas
  + Bases de datos
  + Archivos



**Algoritmos de ordenamiento de datos**

Son algoritmos que fueron realizados para ordenar un conjunto de datos. A lo largo del tiempo se han diseñado numerosas técnicas para lograr el algoritmo más eficiente que logre ordenar una lista de la manera más rápida y eficiente.

Mediante estas características podemos los algoritmos:

* **Complejidad computacional:** Es el mejor, promedio y peor comportamiento que tiene un determinado algoritmo dependiendo del tamaño de la lista. La cantidad de pasos a realizar dependerá en muchos casos del nivel de desorden inicial de la lista, por lo que son necesarios estos 3 valores para comparar el rendimiento de los diferentes algoritmos.
* **Uso de memoria:** Aunque la mayoría de los algoritmos no utilizan más memoria que un espacio extra que el ocupado por la lista a ordenar algunos requieren espacio adicional.
* **Estabilidad:** Depende de la forma en que se organicen los datos cuando tienen varias características pero solo se utiliza una para ordenar. Si se mantiene el orden relativo de la lista es considerado un algoritmo estable.

Ejemplo:

1 2 3 4 5 6 4

Estable 1 2 3 4 4 5 6

Inestable 1 3 4 4 5 6

* **Serial:** Realizan operaciones en series. Es decir siguen un proceso y hasta no terminar una actividad comienzan la siguiente.
* **Paralelo:** Existen otros que aprovechan las ventajas del procesamiento paralelo. Estos permiten realizar operaciones en paralelo sobre la lista permitiendo reducir el tiempo requerido.
* **Comparativos:** Se les llama algoritmos comparativos a aquellos que comparan dos elementos a la vez utilizando un operador de comparación.

**>** mayor que

**>=** mayor o igual que

**==** igual que

**<** menor que

**<=** menor o igual que

**!=** diferente de

* **No comparativos:** Otros tipos de algoritmos como los de ordenamiento entero utilizan operaciones aritméticas sobre las claves para obtener el ordenamiento.
* **Metodo:** El metodo es el proceso elegido para lograr el ordenamiento de la lista:
  + Inserción
  + Intercambio o burbuja
  + Selección
  + Unión
  + Particionado

**Tipos de ordenamiento de datos**

* **Ordenamiento por inserción**

Su funcionamiento consiste en el recorrido por la lista seleccionando en cada inserción un valor como clave y se compara con cada número a su izquierda hasta encontrar uno que sea menor o llegar al principio de la lista.



**Estabilidad**: Estable

**Método**: Inserción

**Comparativo**: Si

**Uso de memoria**: 1

**Complejidad computacional:**

Mejor caso: *O(n)*

Caso promedio: *O(n2)*

Peor caso: *O(n2)*

* **Características del Ordenamiento por INSERCIÓN**
* Inicia por el 2º
* Se compara con el de la ←
* Se compara con TODOS los valores a la ← hasta encontrar un menos o llegar al principio

| 5 | 2 | 4 | 1 | 3 |
| --- | --- | --- | --- | --- |

| 2 | 5 | 4 | 1 | 3 |
| --- | --- | --- | --- | --- |

| 2 | 4 | 5 | 1 | 3 |
| --- | --- | --- | --- | --- |

| 1 | 2 | 4 | 5 | 3 |
| --- | --- | --- | --- | --- |

| 1 | 2 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- |

* **Ordenamiento burbuja**

El algoritmo de ordenamiento de burbuja es uno de los algoritmos de ordenamiento más sencillos aunque no es el más eficiente. Consiste en comparar uno a uno los elementos si el primero es mayot que el segundo se cambian y así sucesivamente hasta llegar al final quedando ordenado el último número. En la siguiente iteración no es considerado este elemento

**Estabilidad**: Estable

**Método**: Intercambio

**Comparativo**: Si

**Uso de memoria**: 1

**Complejidad computacional:**

Mejor caso: *O(n)*

Caso promedio: *O(n2)*

Peor caso: *O(n2)*

* **Características del Ordenamiento de la BURBUJA**
* Inicia con el mayor
* Empieza de →
* Después de la 1ª interacción el número + grande queda ordenado
* 1 cambio a la vez

| 5 | 2 | 4 | 1 | 3 |
| --- | --- | --- | --- | --- |

| 2 | 5 | 4 | 1 | 3 |
| --- | --- | --- | --- | --- |

| 2 | 4 | 5 | 1 | 3 |
| --- | --- | --- | --- | --- |

| 2 | 4 | 1 | 5 | 3 |
| --- | --- | --- | --- | --- |

| 2 | 4 | 1 | 3 | 5 |
| --- | --- | --- | --- | --- |

| 2 | 1 | 4 | 3 | 5 |
| --- | --- | --- | --- | --- |

| 2 | 1 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- |

| 1 | 2 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- |

| 1 | 2 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- |

**UNIDAD 2**

**Tipos abstractos de datos**

Debido al proceso de abstracción, la programación de TAD deberá realizarse siguiendo tres pasos fundamentales:

1. Análisis de datos y operaciones.
2. Especificación del tipo de datos abstracto.
3. Implementación

**Estructuras de datos lineales**

Las estructuras de datos son aquellas en las que los elementos ocupan lugares sucesivos en la estructura y cada uno de ellos tiene un único sucesor y un único predecesor, es decir, sus elementos están ubicados uno al lado del otro relacionados en forma lineal.

Hay 3 tipos de estructuras de datos lineales

* **Listas enlazadas**: las listas enlazadas se constituyen con elementos que están ubicados en una secuencia. Aquí, cada elemento se conecta con el siguiente a través de un enlace que contiene la posición del siguiente elemento. de este modo, teniendo la referencia del principio de la lista podemos acceder a todos los elementos de la misma.

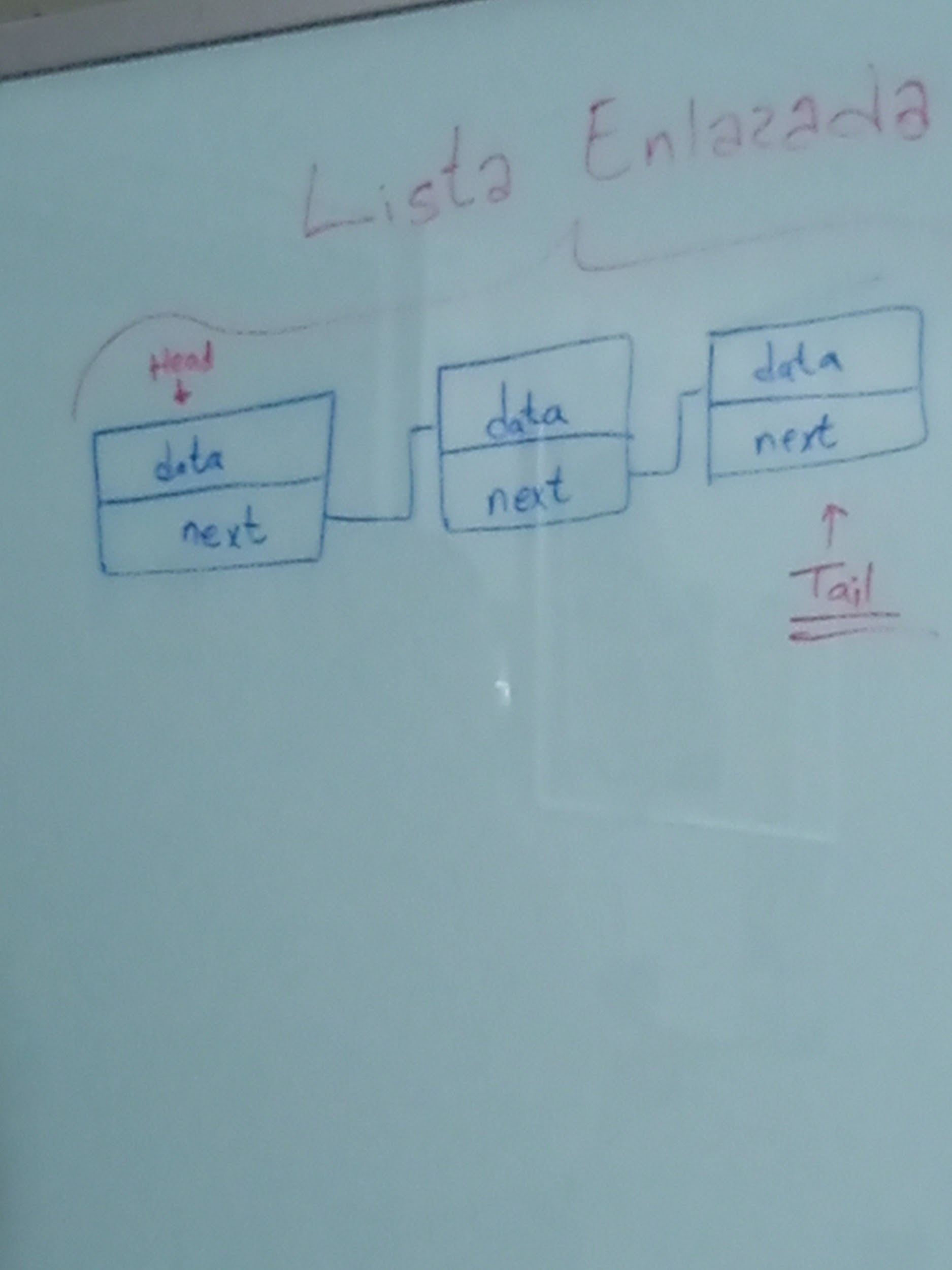
(Ejemplo imagen)

* Es una conexión de nodos contiene un campo puntero, campo información que es lo que conecta un nodo con otro.

Inicio → Dato siguiente → Dato siguiente → Dato siguiente → Fin / Null

* Un nodo contiene un campo de información y un campo puntero es lo que conecta un nodo con otro
* el último nodo de una lista enlazada su sucesor pasa a nulo, lo que referencia que es el único nodo en la lista.
* los nodos pueden tener datos de cualquier tipo.
* se agrega por medio de una función dentro del programa.
* Operaciones básicas de las listas:
  + insertar
  + eliminar
  + que esté vacía

\*Se componen de nodos que tienen dos atributos.\*



* **Pilas:** La pila es un tipo especial de lista lineal dentro de las estructuras de datos dinámicas que permite almacenar y recuperar datos, siendo el modo de acceso a sus elementos de tipo LIFO (de inglés Last In, First Out, es decir, último en entrar, primero en salir).

¿Cómo funciona? A través de dos operaciones básicas: apilar (push), que coloca un objeto en la pila, y su operación inversa, desapilar (pop), que retira el último elemento apilado.

\*Solamente pueden eliminar o insertar en la cima de la lista. FILO (first in last out)\*

* **Colas/Fila:** Una cola es una estructura de datos que almacena elementos en una lista y permite acceder a los datos por uno de los dos extremos de la lista.

Un elemento se inserta en la cola (parte final) de la lista y se suprime o elimina por la frente (parte inicial, cabeza) de la lista.

Los elementos se eliminan (se quitan) de la cola en el mismo orden en que se almacenan y, por consiguiente, una cola es una estructura de tipo FIFO (First- In-First-Out, primero en entrar, Primero en salir o bien primero en llegar/primero en ser servido)

\*Permite emular el comportamiento de una fila o cola de la vida real. FIFO (first in first out)\*

