Resumen

Estructura de Datos

**Tipo de Dato** vs **Estructura de Datos** vs **Tipo de Dato Abstracto (TDA)**

Antes de comenzar a estudiar las estructuras de datos, es necesario que definamos claramente la diferencia entre **tipo de dato**, **estructura de datos** y **tipo de datos abstracto**.

* El **tipo de dato** de una variable es el conjunto de valores que la variable puede tomar. Por ejemplo, una variable de tipo booleana sólo puede tomar un valor verdadero (true) o falso (false).
* Una **estructura de datos** es una colección de variables, posiblemente de diferentes tipos de datos, conectados de varias formas. El contenido básico de una estructura de datos es la celda o nodo, que es capaz de almacenar un dato de un tipo determinado. ***Una estructura de datos define y organiza la interrelación de los datos y las operaciones que se pueden realizar sobre ellos.***
* Un **tipo de datos abstracto (TDA)** es un modelo matemático, junto con varias operaciones definidas sobre ese modelo. Es decir, un TDA define un modelo de un tipo de datos, indicando su posible contenido y funcionalidad (operaciones), pero no dice cómo debe implementarse.

Como programadores, lo ideal es diseñar algoritmos en términos de TDAs. Luego, al momento de implementar un algoritmo en un lenguaje de programación particular, será necesario encontrar la manera de representar los TDAs con los tipos de datos y operadores de dicho lenguaje de programación.

**Clasificación de las Estructura de Datos**

Las estructuras de datos son modelos para organizar y acceder a un conjunto de datos del mismo tipo. Existen distintos tipos de estructura de datos y cada estructura tiene propósitos diferentes y la elección de las estructuras de datos adecuadas a un problema es uno de los puntos clave de un desarrollo de software.

* **Estructuras Lineales:** Guardan los datos (elementos) como si estuvieran en una fila (uno detrás de otro). Visualmente las estructuras lineales tienen la forma que se muestran en la Figura 1.1. Pueden almacenar elementos repetidos. Tienen operaciones para añadir y quitar elementos de la estructura y también para preguntar por algún elemento en particular. Un ejemplo de estructura lineal es el **arreglo**.

Diagrama

Descripción generada automáticamente con confianza media

* **Estructuras jerárquicas:** Los árboles son estructuras de datos jerarquizadas. Cada árbol está constituido por una colección de elementos llamados nodos, uno de los cuales se denomina **raíz**. Cada nodo tiene una relación de parentesco dentro de la estructura jerárquica (de padres a hijos). Hay distintos tipos de árboles según la cantidad máxima de hijos que puede tener un nodo, si sus elementos se mantienen ordenados o no, etc.

Icono

Descripción generada automáticamente con confianza baja

* **Grafos:** son estructuras de datos en las que se pueden expresar relaciones de conexión sin restricción entre los diversos elementos. Estos elementos son denominados vértices o nodos. Los vértices están unidos por líneas llamadas aristas o arcos. Uno de los problemas más comunes que se busca resolver al implementar grafos, es encontrar el camino de costo mínimo o verificar si dos nodos tienen un camino entre sí.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

**Definición e implementación de un TDA**

Al trabajar con estructuras de datos se utiliza el concepto de **TDA** tanto para definir a la estructura en si, como para crear y manipular a los datos almacenados en ellas.

Un **Tipo de Dato Abstracto** representa un conjunto de datos que cumplen ciertas condiciones especificadas por el mismo TDA, mas un conjunto de operaciones primitivas que representan el comportamiento de este. Es decir que el **TDA** tiene una **identidad** (información) y un **comportamiento** (operaciones), con el cual permite la interacción o comunicación consigo mismo. **El TDA hace abstracción de la implementación al momento de definirlo y también cuando este es utilizado:**

**Abstracción al utilizar un TDA:** Al usar un TDA, solo se debe saber de que se trata dicho TDA, es decir que tipos de datos involucra y cuales son las operaciones que ofrece. La parte que se da a conocer se llama la **INTERFAZ** y esta conformada por los nombres, parámetros y resultados de todas sus operaciones o funciones. Como dichas operaciones han sido construidas y con que variables, constantes o estructuras internan cuenta, conforman la IMPLEMENTACION DEL TDA y siempre son transparente al usuario. **ESTO QUIERE DECIR QUE SE MANTIENEN OCULTAS Y EL USUARIO FINAL NO TIENE CONOCIMIENTO DE COMO SE IMPLEMENTAN**

**Abstracción al definir e implementar un TDA:** Un TDA es ante todo un producto software, cuyo usuario será otro programador. Al igual que un programa común, al archivo o archivos necesarios para que el TDA funcione (IMPLEMENTACION) debe adjuntarse la documentación apropiada (INTERFAZ + comentarios de uso). Además es muy importante que al construir un TDA se comience por tener un buen diseño previo. Para ello se recomienda lo siguiente:

1. Analizar el problema a plantear y obtener una lista de sus características principales.
2. Hacer un diagrama analizando la identidad (valores variables y/o constantes, tipos de datos de cada una) y sus operaciones (funciones, procedimientos)
3. Implementarlo en el lenguaje deseado, haciendo uso de buenas pautas de programación y documentación.
4. Testear el TDA una vez terminado.
5. Volver a rever y ajustar el producto de los pasos anteriores mientras sea necesario.

**Recomendaciones para diseño e implementación de TDA**

* Un TDA tiene que modelar o estar enfocado en **un solo problema cada uno, nunca varios a la vez.**
* Las operaciones de un TDA (**primitivas**) deber ser **implementadas** con **códigos cortos.** Siempre se debe **MODULARIZAR** con operaciones ocultas (privadas del TDA).
* Todo TDA debe tener una **primitiva** para su “construcción” y, si el lenguaje lo requiere, otra para su “destrucción”.
  + Todo TDA que tenga clave univoca no puede tener el constructor vacío. Debe existir al menos un constructor a partir de su clave. Luego pueden agregarse otros constructores con la clave y otros datos.
  + En Java, no es necesaria una operación de destrucción dado que utiliza un recolector de basura que se dispara automáticamente y no tiene primitivas para devolver el espacio liberado a la memoria heap.
* La información de la entidad (datos) debe estar siempre oculta, excepto por medio de las operaciones que correspondan. Para ello, al implementar en Java, los atributos deben ser siempre privados (private). Además, las operaciones de recuperación (get) y modificación (set) de dichos atributos deben ser diseñadas sin ir en contra de la encapsulación y de la naturaleza del TDA que representan:
  + No debe haber método *set* para los campos clave de un TDA.
  + No debería haber *métodos set* para cualquier campo considerado **inmutable.**
  + Hay campos para los cuales la operación obtener *get* debería estar vedada. Por ejemplo el campo contraseña de un usuario.
* Las primitivas del TDA tienen que permitirle al usuario manipularlo sin dar información sobre la implementación del TDA (ocultamiento de información). El ocultamiento de información al implementar las operaciones primitivas es fundamental.
* El testeo del TDA tiene que ser 100% exitoso.