**ADT (*Abstract Data Type*: Tipo Abstracto de Datos)**

Un tipo abstracto de datos se refiere a una especificación matemática de un conjunto de datos (valores) y las operaciones que pueden realizarse sobre ellos, independientemente de su representación interna o implementación concreta.

En lugar de enfocarse en los detalles de representación y cómo se implementan las operaciones, un ADT define una interfaz o describe una serie de operaciones claramente establecidas que pueden realizarse sobre los datos. Los ADT permiten abstraer los detalles de implementación y facilitan el diseño y la comprensión de programas al separar la lógica del programa de los detalles de implementación.

Una **clase** es una **implementación concreta** de un ADT. La clase define cómo se implementan las operaciones y cómo se almacenan los datos del ADT en la memoria. Así, una clase es una definición o modelo que establece las propiedades (atributos) y comportamientos (métodos) de los objetos concretos que se van a crear a partir de ella.

ADT = valores + operaciones

Un **tipo elemental de datos**, también conocido como **tipo primitivo** o tipo básico, es un tipo de dato predefinido por el lenguaje de programación que representa un valor simple y no puede descomponerse en partes más pequeñas. Los tipos primitivos están muchas veces restringidos por el *hardware* o la implementación de un lenguaje concreto de programación. Están diseñados para contener valores individuales o atómicos y suelen ser los componentes fundamentales para construir estructuras de datos más complejas.

Casi siempre los tipos primitivos corresponden con números enteros o valores de un subconjunto propio de los números enteros. A menudo, los tipos se definen como un mapeo de un conjunto de símbolos sobre el conjunto de número enteros (codificación).

Almacenamiento contiguo vs enlazado

Acceso secuencial vs. directo

Estructuras lineales vs. No lineales

Estructiras asociatiovas vs. No asociativas

Representación de hielras

p-string

c-string

asignación de hileras en c

Las estructuras de datos **homogéneas** son aquellas que están compuestas por elementos del mismo tipo de dato. En otras palabras, todos los elementos almacenados en una estructura de datos homogénea son del mismo tipo y **tienen la misma longitud o tamaño en memoria**. En un lenguaje de programación moderno, es común que solamente los tipos de datos primitivos (elementales) sean considerados en una estructura homogénea. Sin embargo, el uso de referencias o apuntadores permite abstraer los detalles de almacenamiento cuando se trata de estructuras no primitivas (instancias de clases).

Los flujos estándar son canales de comunicación en tiempo real entre un programa y su entorno, que permiten la entrada y salida de datos a través de la consola o terminal en la que se está ejecutando el programa. En los sistemas operativos Unix y Windows, existen tres flujos estándar: entrada estándar (stdin), salida estándar (stdout) y salida de errores estándar (stderr). En los diferentes lenguajes de programación, se utilizan nombres específicos para cada flujo. **En C++, por ejemplo, los flujos estándar se definen en el archivo de cabecera <iostream> y se llaman cin, cout y cerr, respectivamente.**

La entrada estándar (stdin) es el canal de entrada a través del cual un programa recibe datos desde la consola o terminal. Por ejemplo, si se está ejecutando un programa de línea de comandos y se le solicita al usuario que ingrese información, la entrada estándar será el medio por el cual el programa recibe la entrada del usuario.

La salida estándar (stdout) es el canal de salida a través del cual un programa envía datos a la consola o terminal. Por ejemplo, si se está ejecutando un programa que produce una lista de resultados, la salida estándar será el medio por el cual el programa muestra la lista de resultados al usuario.

La salida de errores estándar (stderr) es similar a la salida estándar, pero se utiliza específicamente para mostrar mensajes de error o advertencia. Por ejemplo, si un programa encuentra un error mientras se está ejecutando, puede mostrar un mensaje de error en la salida de errores estándar para indicar al usuario que algo salió mal.

En los sistemas operativos Unix y Windows, se pueden redirigir los flujos estándar utilizando símbolos especiales en la línea de comandos. Estos símbolos indican al sistema operativo que se deben redirigir los flujos estándar de entrada, salida y errores a diferentes ubicaciones o dispositivos.

Los símbolos para redirigir los flujos estándar (en Windows o en UNIX) son los siguientes:

**“<“** se utiliza para **redirigir la entrada estándar** desde un archivo. Por ejemplo, el comando “sort < archivo.txt” ordenará el contenido del archivo “archivo.txt”.

**“>“** se utiliza para **redirigir la salida estándar** a un archivo. Por ejemplo, el comando “dir > archivo.txt” redirigirá la salida del comando “dir” al archivo “archivo.txt” en lugar de mostrarla en la pantalla.

**“>>“** se utiliza para **redirigir la salida estándar y agregarla (*append*)** a un archivo existente. Por ejemplo, el comando “echo “texto” >> archivo.txt” agregará el texto “texto” al final del archivo “archivo.txt”.

“2>“ se utiliza para **redirigir la salida de errores estándar** a un archivo. Por ejemplo, el comando “dir noexiste 2> errores.txt” redirigirá los errores del comando “dir” al archivo “errores.txt” en lugar de mostrarlos en la pantalla.

En Windows, además, también se utiliza “&>” para redirigir la salida de errores estándar a un archivo.