Instituto Politécnico Nacional

Escuela Superior de Cómputo

Estructuras de Datos

Tema 01: Abstracción en los lenguajes de programación y tipo abstracto de dato (TAD)

M. en C. Edgardo Adrián Franco Martínez
http://www.eafranco.com
edfrancom@ipn.mx
@edfrancom *edgardoadrianfrancom*

Contenido

- Abstracción (Concepto)
  - Introducción
  - Abstracción (Comprensión por descomposición)
- Mecanismos de abstracción en programación
  - Por parametrización
  - Por especificación
- Tipos de abstracción
  - Abstracción Procedimental
  - Abstracción de Iteración
  - Abstracción de datos
- Tipo abstracto de dato (TAD)
  - Visiones de un TAD
  - Separación de la interfaz e implementación
  - Caracterización
  - Operaciones sobre un TAD
  - Especificación genérica de un TAD
- Especificación de los TAD
  - Especificación informal o genérica
  - Especificación formal
    - Método axiomático o algebraico
    - Método constructivo u operacional
- Estructura de la especificación para los TAD's en el curso

Abstracción (Concepto)

- **Abstracción:** La abstracción (del latín abstrahere, 'alejar, sustraer, separar') es una operación mental destinada a aislar conceptualmente una propiedad o función concreta de un objeto, y pensar o hablar qué es, mentalmente sobre ésta, ignorando mentalmente otras propiedades del objeto en cuestión.

- ***"En la abstracción convertimos la realidad en una idea que podemos manejar de acuerdo a nuestra comprensión"***

Introducción

- **Abstracción:** Operación intelectual que ignora selectivamente partes de un todo para facilitar su comprensión.

- **Abstracción en la resolución de problemas:** Ignorar detalles específicos buscando generalidades que ofrezcan una perspectiva distinta, mas favorable a su resolución, i.e. es una descomposición en que se varía el nivel de detalle.

La abstracción sirve para:

Problema bajo un contexto → Representación detallada y modularizada bajo otro contexto

Abstracción (Comprensión por descomposición)

- **Propiedades de una descomposición útil:**
  - Todas las partes deben estar al mismo nivel.
  - Cada parte debe poder ser abordada por separado.
  - La solución de cada parte debe poder unirse al resto para obtener la solución final.

Mecanismos de abstracción en programación

- ***Abstracción por parametrización:*** Se introducen parámetros para abstraer un numero infinito de posibles computaciones.
  - Ejemplo: calculo de ***cos β***

- ***Abstracción por especificación:*** Permite abstraerse de la implementación concreta de un procedimiento asociándole una descripción precisa de su comportamiento.
  - Ejemplo: ***double sqrt (double a);***
    - Requisitos: a >0;
    - Efecto: devuelve una aproximación de $\sqrt{a}$

  - *La especificación es un comentario lo suficientemente definido y explicito como para poder usar el procedimiento sin necesitar conocer otros elementos.*

Abstracción por parametrización

- En una **abstracción por parametrización** se obtienen las *abstracciones procedimentales*

  - La definición de parámetros nos permite abstraer su valor concreto (actual).

  - También abstraemos la particularidad de la ejecución concreta.

calculo de ***cos β***

Abstracción por especificación

Una abstracción por especificación, se suele expresar en términos de:

- **Precondición:** Condiciones necesarias y suficientes para que el procedimiento se comporte como se prevé.

- **Postcondición:** Enunciados que se suponen ciertos tras la ejecución del procedimiento, si se cumplió la precondición.


```
int busca_minimo(float * array, int num_elem)
```

/***

precondicion:

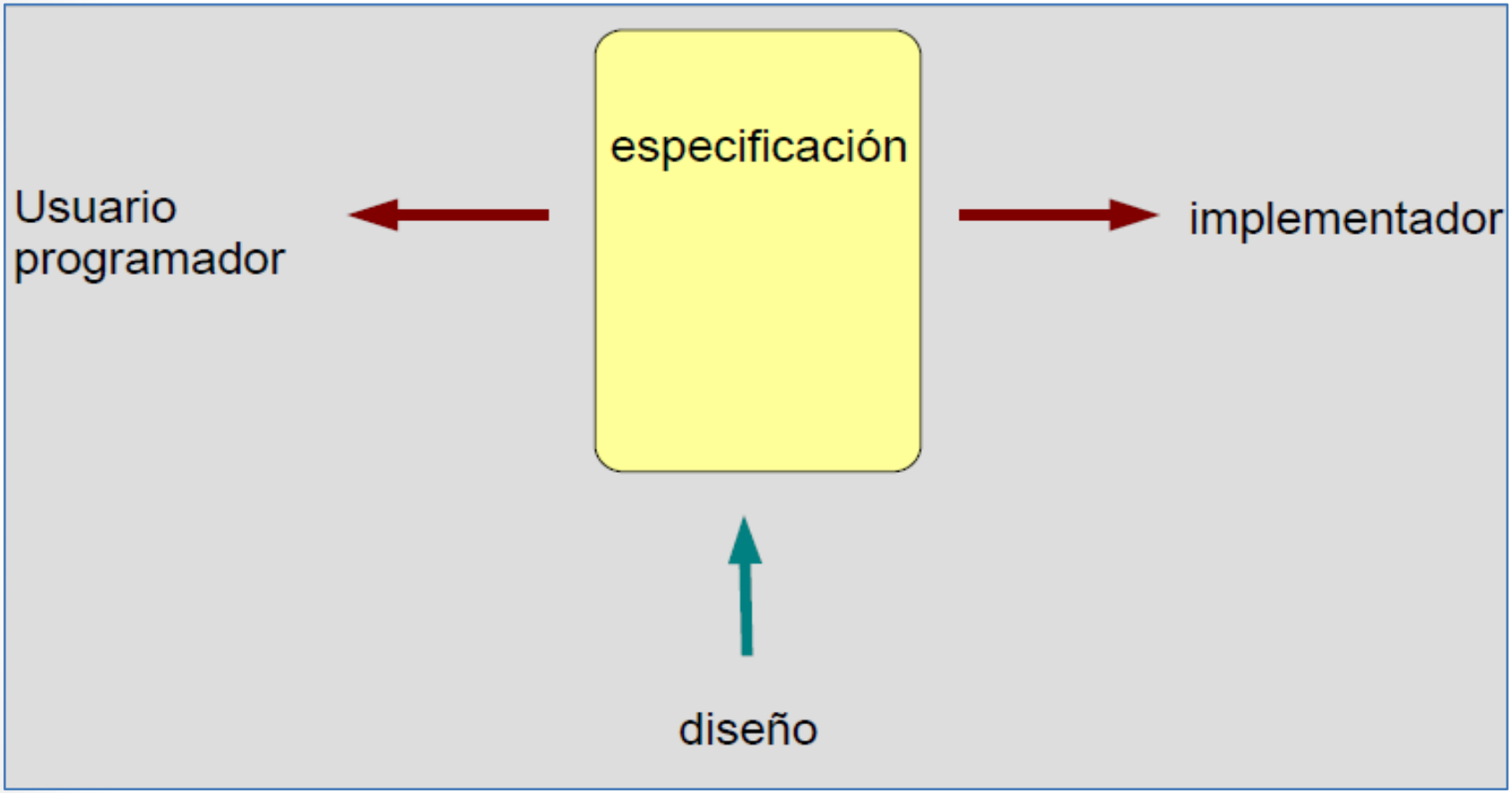
- num_elem > 0.
- 'array' es un vector con 'num_elem' componentes.

postcondicion:

devuelve la posición del mínimo elemento de 'array'.

***/

Esquema de abstracción por especificación



Tipos de abstracción

- **Abstracción Procedimental:** Definimos un conjunto de operaciones *(procedimiento)* que se comporta como una operación.

- **Abstracción de Iteración:** Abstracción que permite trabajar sobre colecciones de objetos sin tener que preocuparse por la forma concreta en que se organizan.

- **Abstracción de Datos:** Tenemos un conjunto de datos y un conjunto de operaciones que caracterizan el comportamiento del conjunto. Las operaciones están vinculadas al tipo de abstracción del dato.

Abstracción procedimental

- Permite abstraer un **conjunto preciso de operaciones de computo como una operación simple**. Realiza la aplicación de un conjunto de entradas en las salidas con posible modificación de entradas.

- La identidad de los datos no es relevante para el diseño. Solo interesa el numero de parámetros y su tipo.

- Al especificar una abstracción procedimental es irrelevante la implementación, pero no que hace.

  - **Localidad:** Para implementar una abstracción procedimental no es necesario conocer la implementación de otras que se usen, solo su especificación.

  - **Modificabilidad:** Se puede cambiar la implementación de una abstracción procedimental sin afectar a otras abstracciones que la usen, siempre y cuando no cambie la especificación.

Abstracción de iteración

- **La abstracción de iteración y los iteradores:** Los iteradores son una generalización del mecanismo de iteración disponible en la mayoría de los lenguajes de programación. Éstos permiten a los usuarios iterar sobre los tipos de datos arbitrarios de un modo práctico y eficaz.
  - *E.g. llevar a cabo alguna operación para cada uno de los elementos de un arreglo.*
- Para todos elementos del conjunto -> hacer acción

Abstracción de datos

- **Abstraer datos** se refiere a especificar un *conjunto de datos* y *operaciones* que caracterizan el comportamiento del conjunto. A todo el **proceso** de extraer, definir, implementar y especificar datos es a lo que llamamos ***Abstracción de Datos.***

- **Abstracción de datos implica:**
  - **Especificar:** Descripción del comportamiento del TAD.
  - **Representar:** Forma concreta en que se representan los datos en un lenguaje de programación para poder manipularlos.
  - **Implementar:** La forma especifica en que se expresan las operaciones.

Abstracciones de datos

- **Tipo de datos:** proporcionados por los leguajes de alto nivel. La representación usada es invisible al programador, al cual solo se le permite ver las operaciones predefinidas para cada tipo.

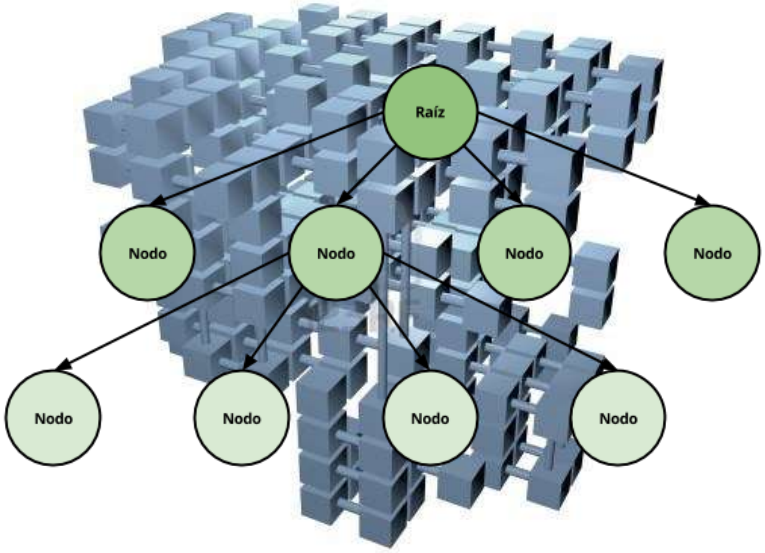
- **Tipos definidos por el programador:** que posibilitan la definición de valores de datos más cercanos al problema que se pretende resolver. *(p.g. definir estructuras en C).*

- **Tipo abstracto de dato (TAD):** para la definición y representación de tipos de datos *(valores + operaciones)*, junto con sus propiedades.
- A los TAD comúnmente se les conoce con el nombre de
- **Objetos:** Son TAD a los que se añade propiedades de reutilización y de compartición de código.

Tipo Abstracto de Dato (TAD)

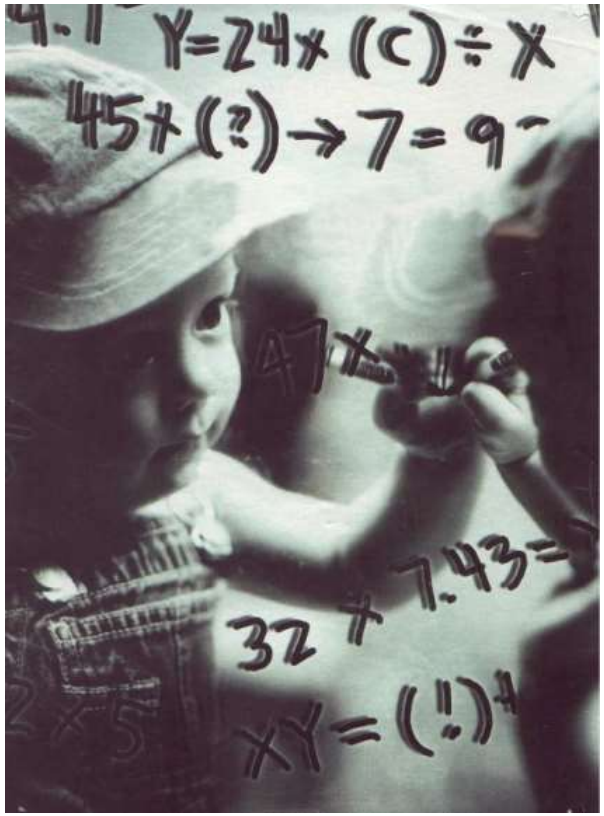
"TAD y Abstracción de Datos no son lo mismo"

- **Tipo Abstracto de Dato (TAD):** Entidad abstracta formada por un conjunto de datos organizados en cierta estructura y una colección de operaciones asociadas especificados formalmente.



- Estrictamente un **tipo de dato abstracto *(TDA)*** o **tipo abstracto de datos *(TAD)*** es un **modelo matemático** compuesto por una **colección de operaciones definidas** sobre un conjunto de datos para el modelo.

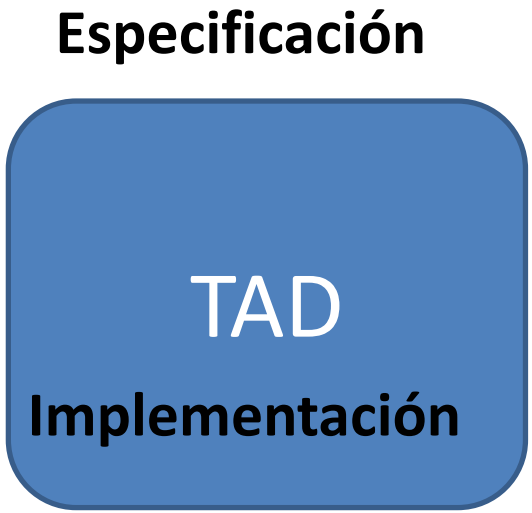
$$
\begin{aligned}
&\langle \mathbf{S}_k \rangle \\
&= \langle \alpha | \mathbf{S}_k | \alpha \rangle \\
&= \langle \alpha | \mathbf{I} \cdot | \mathbf{S}_k | \cdot \mathbf{I} | \alpha \rangle \\
&= \langle \alpha | \sum_{a'=+,-} |a'\rangle\langle a'| \cdot |\mathbf{S}_k| \cdot \sum_{a''=+,-} |a''\rangle\langle a''| \alpha \rangle \\
&= \sum_{a'=+,-} \sum_{a''=+,-} \langle \alpha | |a'\rangle\langle a'| |\mathbf{S}_k| |a''\rangle\langle a''| \alpha \rangle \\
&= \frac{\hbar}{2} \sum_{a'=+,-} \sum_{a''=+,-} \langle \alpha | |a'\rangle\langle a'| |\sigma_k| |a''\rangle\langle a''| \alpha \rangle \\
&= \frac{\hbar}{2} \begin{bmatrix} \langle \alpha | + \rangle & \langle \alpha | - \rangle \end{bmatrix} \sigma_k \begin{bmatrix} \langle + | \alpha \rangle \\ \langle - | \alpha \rangle \end{bmatrix} \\
&= \frac{\hbar}{2} \chi^\dagger \sigma_k \chi
\end{aligned}
$$



Visiones de un TAD

- Hay dos visiones de un TAD:
  - ***Visión externa:*** especificación.
  - ***Visión interna:*** representación e implementación.

- Ventajas de la separación:
  - Se puede cambiar la visión interna sin afectar a la externa.
  - Facilita la labor del programador permitiéndole concentrarse en cada fase por separado.



Separación de la interfaz e implementación

- Cuando se usa en un programa de computación, **un TAD es representado por su interfaz**, la cual sirve como cubierta a la correspondiente implementación.

- Los usuarios de un TAD tienen que preocuparse por la interfaz, pero no por la implementación. Esto se basa en el **concepto de ocultación de información** y proporciona una protección para el programa.

- La solidez de un TAD reposa en la idea de que la **implementación está escondida al usuario**. Solo la **interfaz es pública**, i.e. *el TAD puede ser implementado de diferentes formas, pero mientras se mantenga consistente con la interfaz, los programas que lo usan no se ven afectados.*

Caracterización

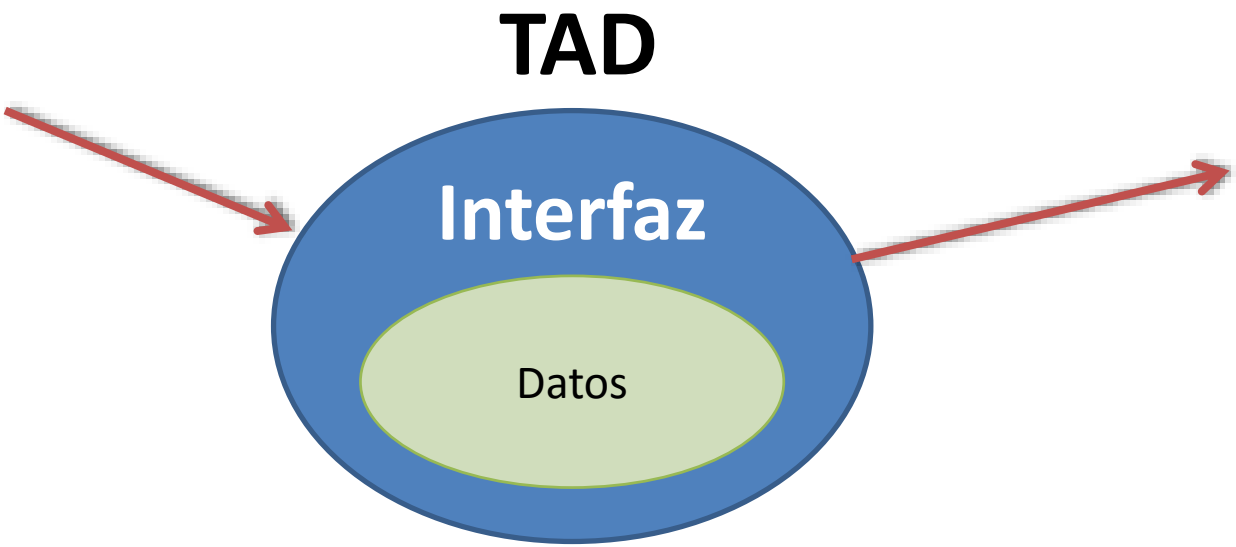
- Cuando hablemos de un TAD **no se hace ninguna alusión al tipo de los elementos** sino tan sólo **a la forma en que están dispuestos estos elementos**. **Sólo nos interesa la estructura** que soporta la información **y sus operaciones**. Para determinar el comportamiento estructural basta con observar la conducta que seguirán los datos.

- Un TAD 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:

  - La maquinaria algorítmica que implemente, la semántica de las operaciones

  - Los datos que sirvan de enlace entre los elementos del TAD.

- Un **TAD** representa una abstracción de datos:
  - Se **destacan los detalles** *(normalmente pocos)* de la **especificación** *(el qué).*
  - Se **ocultan los detalles** *(casi siempre numerosos)* de la **implementación** *(el cómo).*



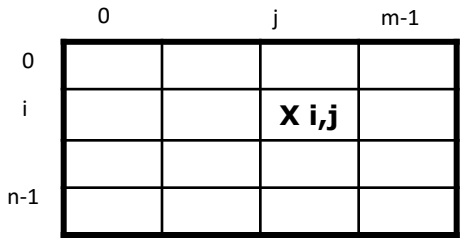
Operaciones sobre un TAD

- **Constructora:** Crea elementos del TAD.
- **Modificadora:** Permite alterar el estado de un elemento del TAD.
- **Analizadora:** Permite consultar por el estado del objeto y retornar algún tipo de información.
- **Persistencia:** Permiten almacenar un TAD indefinidamente.
  - P.g. Analizadoras:
    - Comparación de igualdad entre objetos.
    - Salida en pantalla, permite visualizar el estado de un elemento del TAD (sirve como base para alguna interfaz o depuración en pruebas).
  - P.g. Modificadoras:
    - Copiar un elemento por otro, cambiando su estado.
    - Destrucción, retorna el espacio de memoria dinámica ocupada por un elemento.

Especificación genérica de un TAD

- **Nombre del TAD**
- **Representación abstracta**
- **Restricciones**
- **Lista de Operaciones**
- **Definición de cada operación**

- **P.g.**
  - **TAD Matriz**
  - **Representación abstracta:**



  - **Restricciones:** n>0, m>0
  - **Lista de Operaciones:**
    - CrearMatriz(i, j)
    - AsignarMatriz(M, i, j, valor)
    - ObtenerDatoMatriz(M, i, j)
    - SumaMatriz(M1, M2)
    - MatrizNegativa(M)
    - RestarMatriz(M1, M2)
    - MatrizInversa(M)
    - MostrarMatriz(M)

Especificación de los TAD's

- **Especificar:** Determinar, explicar algo con todos los detalles precisos para su identificación o entendimiento.

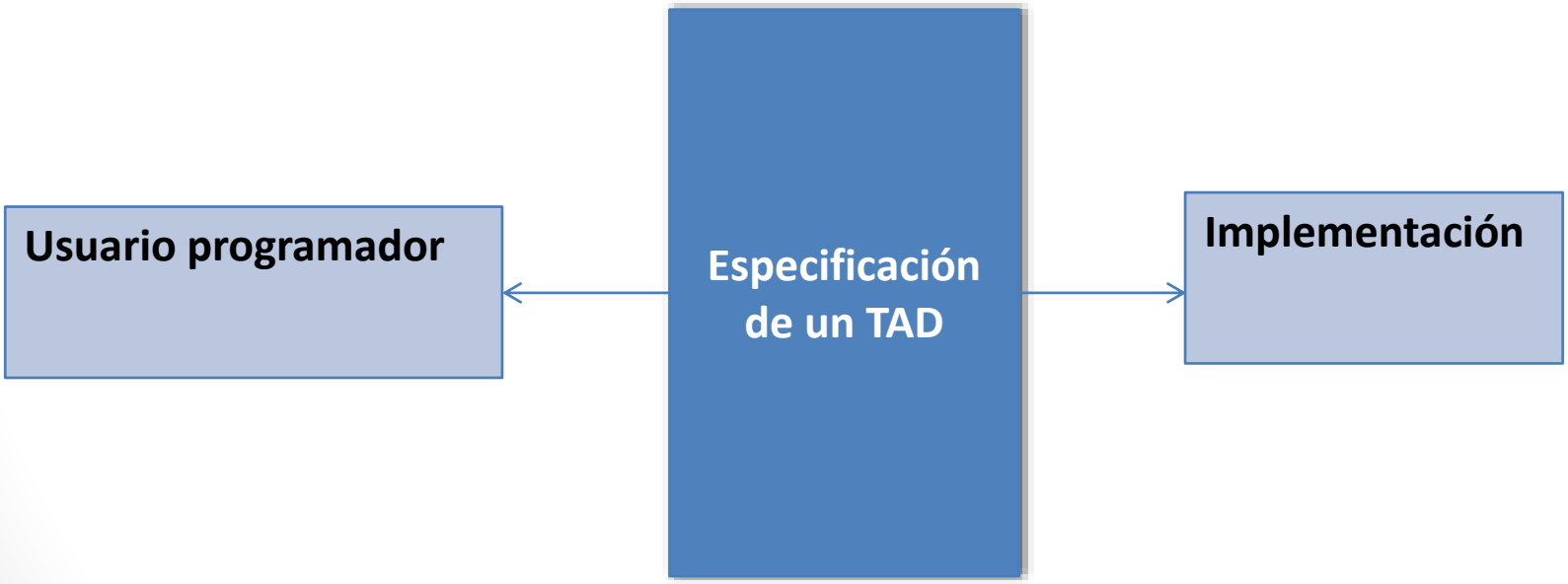
Una mala especificación de la interfaz

Característica Eléctrica	Corriente	AC120V / AC230V
	Consumo	60 HZ / 50 HZ
	Frecuencia	MA x 230W
	Temperatura del uso	30°C ~ 65°C
Dimensiones	Cama superior	1200 x 700 x 70
	Cama inferior	800 x 700 x 170
	Altura	1900 x 640 x 400
	Volumen de cama	2000 x 700 x 560
Peso	Cama superior	39,5 kg
	Cama inferior	20 kg
	Base	29 kg
	Generador de calor	88 kg

Especificación técnica de un producto

- En una especificación de un TAD, se establecen sus propiedades, se define totalmente su comportamiento, pero no se dice nada sobre su implementación.

- Indica el tipo de entidades que modela, que operaciones se les pueden aplicar, como se usan y que hacen.



Características de la especificación de un TAD

- **Precisa:** Menciona únicamente aquello que es imprescindible.
- **General:** Se adapta a los diferentes contextos que se podrían llegar a manejar.
- **Legible:** Debe ser entendible para lograr una comunicación claro entre el especificador, el implementador y los usuarios del tipo.
- **No ambigua:** Que evite futuros problemas en la manera de interpretarse.

Tipos de especificaciones de un TAD

- Existen dos tipos de especificaciones de una TAD

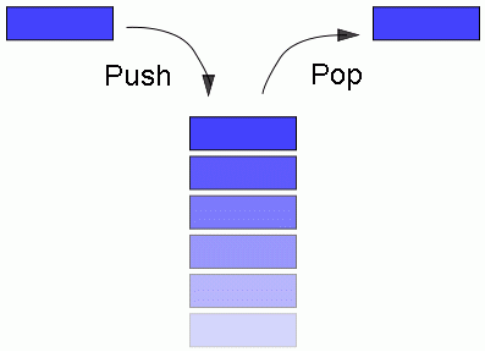
1. Especificación **informal o genérica**
   - Se apoya de diagramas y explicaciones textuales.

2. Especificación **formal**
   - Utiliza un lenguaje formal
   - Modela las operaciones y estructuras formalmente

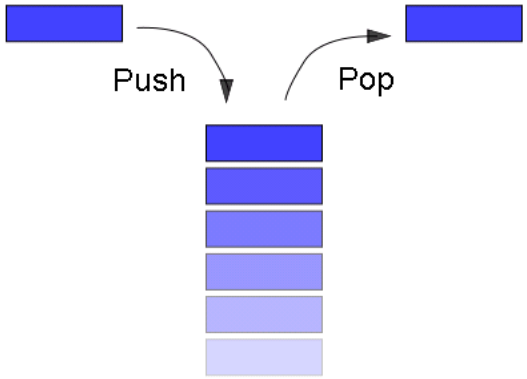
Especificaciones informales o genéricas

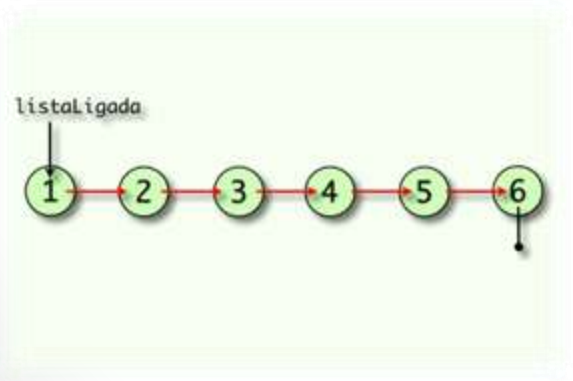
- En este tipo de especificación a veces **se llega a cierta ambigüedad** e imprecisión ya que su descripción se realiza en un lenguaje más natural.

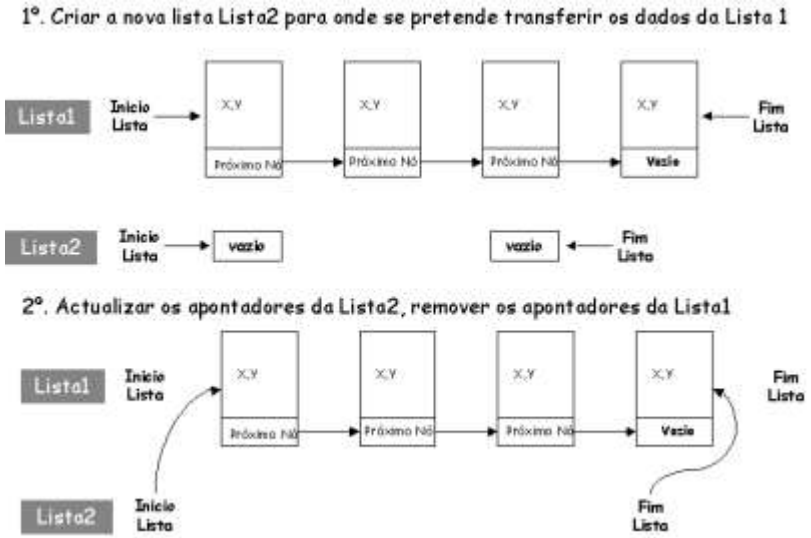
  - Explicación redactada
  - Apoyada en imágenes y diagramas
  - Sencilla y fácil de entender



- Se apoya de una ***explicación en lenguaje natural, imágenes y diagramas,*** donde se mencionan los objetos sobre los que opera el TAD, la estructura del TAD y las posibles operaciones sobre el mismo.







Especificación informal de un TAD
(Ejemplo 1: TAD Número Natural)

- Un número natural es cualquiera de los números que se usan para contar los elementos de un conjunto (el cero es el número de elementos del conjunto vacío).

- El conjunto de los números naturales se representa por $\mathbb{N}$ y corresponde al siguiente conjunto numérico:

$$\mathbb{N} = \{1, 2, 3, 4, 5, 6, 7, \ldots\}$$

- Los números naturales son un conjunto cerrado para las operaciones de la adición y la multiplicación, ya que al operar con cualquiera de sus elementos, resulta siempre un número perteneciente a.

Especificaciones formales

- En este tipo de especificación no debe haber imprecisión. En esta especificación la descripción se realiza mediante reglas algebraicas o expresiones estándares que **no permiten ambigüedad** en la descripción
  - Se expresan en un lenguaje formal
  - Lenguajes algebraicos
  - Compleja

    - Una especificación formal de tipo de datos abstracto consta de cuatro secciones: TIPOS, FUNCIONES, AXIOMAS y PRE-CONDICIONES.

- La especificación formal de un TAD describe un TAD y sus operaciones de manera precisa sin ambigüedades apoyándose en ***lenguajes formales*** que permitan que cualquiera entienda el mismo significado de las operaciones, los datos y las características de un TAD.

- Comúnmente se emplean notaciones formales para definir la **semántica**, ya sea a través de métodos axiomáticos o algebraicos o método constructivos u operacionales.

Notaciones formales para definir la semántica

- Las distintas notaciones formales existentes difieren en la forma de **definir la semántica**:

  - **Método axiomático o algebraico:** Se establece el significado de las operaciones a través de relaciones entre operaciones (axiomas).
    - Significado implícito de las operaciones.

  - **Método constructivo u operacional:** Se define cada operación por sí misma, independientemente de las otras. Significado explícito de las operaciones.

Definición de la semántica
(Método axiomático o algebraico)

- Este tipo de especificación consiste en la determinación del como hay que escribir las operaciones de un TAD, proporcionando el tipo de operandos y el tipo de resultados.
- P.g.
  - *int '+' (int a,b)*
  - *int '-' (int a,b)*
  - *int abs (int a)*

Definición de la semántica
(Método axiomático o algebraico)

- La **notación algebraica** y **ecuaciones** consisten en proporcionar un conjunto de **axiomas y operaciones verificadas** para cada una de las operaciones de un TAD.

- P.g. *Tipo String*.
  - Podemos definir el tipo String como:
  - $\{a \mid a=<a_1, \ldots, a_n>, a_i \text{ es carácter}, n \leq N\}$

Especificación formal de un TAD

Ejemplo 1 Definición de la semántica mediante el método axiomático o algebraico: (TAD Números Naturales)

- **Nombre:** natural
- **Conjuntos:** N conjunto de naturales, B conjunto de valores booleanos.
- **Sintaxis:**
  - cero: → N
  - sucesor: N → N
  - escero: N → B
  - igual: N x N → B
  - suma: N x N → N

- ***Semántica:*** *para todo m y n pertenecientes a N*
  1. *escero (cero) = true*
  2. *escero (sucesor (n)) = false*
  3. *igual (cero, n) = escero (n)*
  4. *igual (sucesor (n), cero) = false*
  5. *igual (sucesor (n), sucesor (m)) = igual (n, m)*
  6. *suma (cero, n) = n*
  7. *suma (sucesor (m), n) = sucesor (suma (m, n))*

Definición de la semántica
(Método constructivo u operacional)

- **Método constructivo u operacional:** En la semántica de este método se establecen las precondiciones y las postcondiciones de las operaciones:

  - **Precondición:** Relación que se debe cumplir con los datos de entrada para que la operación se pueda aplicar.

  - **Postcondición:** Relaciones que se cumplen después de ejecutar la operación. No debe incluir detalles de implementación.

Especificación formal de un TAD

Ejemplo 2 Definición de la semántica mediante el método constructivo u operacional: (TAD Números Naturales)

max_restring: Z x Z → Z

pre-max_restring (x, y) ::= (x > 0) ^ (y > 0)

post-max_restring (x, y; r) ::= (r ≥ x) ^ (r ≥ y) ^ (r=x V r=y)

- *¿Qué sucedería si x o y no son mayores que 0?*
  - No se cumple la precondición y no podríamos asegurar que se cumpla la potscondición.

Estructura de la especificación para los TAD's en el curso

"Para el curso de Estructuras de Datos emplearemos la siguiente estructura de especificación, dentro de la cuál se utilizaran especificaciones tanto formales como informales según sea el TAD a especificar"

1. **Cabecera:** nombre del tipo y listado de las operaciones.

2. **Definición:** Descripción del comportamiento sin indicar la representación. Se debe indicar si el tipo es mutable o no. También se expresa donde residen los datos internos.

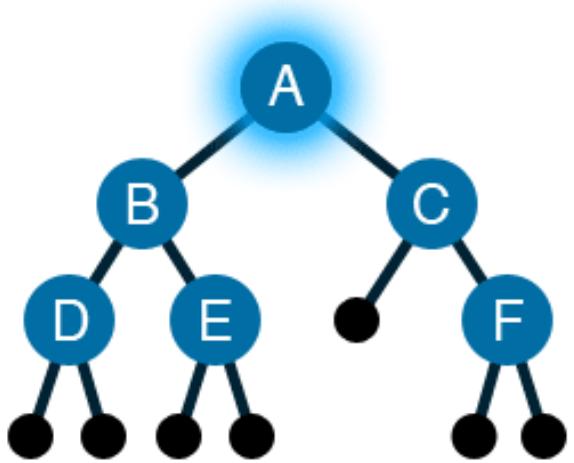
3. **Operaciones:** Especificar las operaciones una por una como abstracciones procedimentales
   - Condición de los parámetros de entrada de las operaciones.
   - Resultados de las operaciones

4. **Observaciones:** Notas importantes a considerar como usuario y como implementador del TAD

Estructuras de Datos

- “Las estructuras de datos son **colecciones de datos**, no necesariamente del mismo tipo, **relacionadas entre sí de alguna forma** y a las que se les asocian reglas de **operaciones sobre los datos”**.

Las estructuras de datos están caracterizadas por el tipo de dato de los elementos guardados en la estructura y por la relación definida sobre estos elementos (Son el resultado de un proceso de abstracción de datos y pueden ser especificadas a través de un TDA formalmente).



Output:

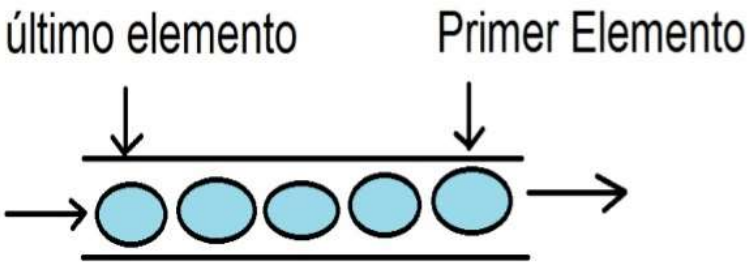
“Una Estructura de Datos es una forma particular de organizar y almacenar datos en una computadora para que puedan ser utilizados de manera eficiente.”

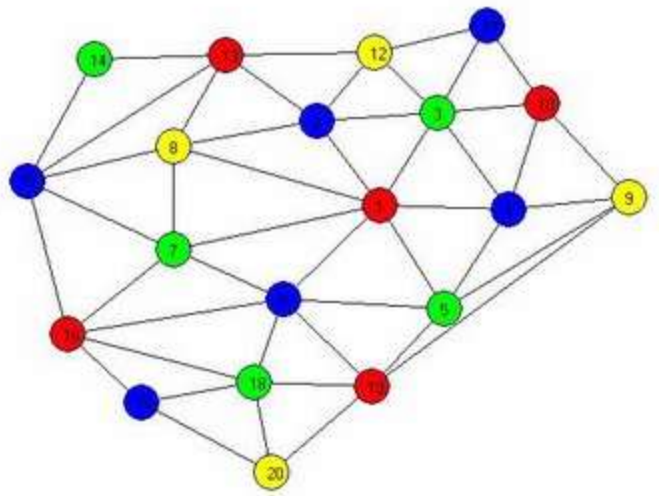
–Puede verse como un “Modelo de organización de los datos”-

- Al nivel de las estructuras de datos son totalmente irrelevantes las operaciones sobre un elemento en particular, solamente tienen carácter relevante las **operaciones que envuelvan la estructura de forma global**.

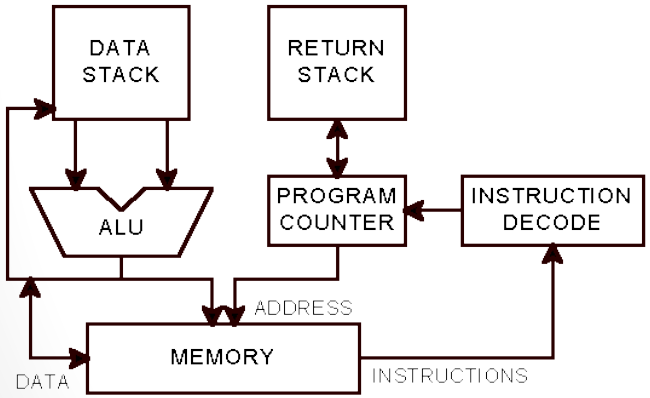
Clasificación de las Estructuras de Datos

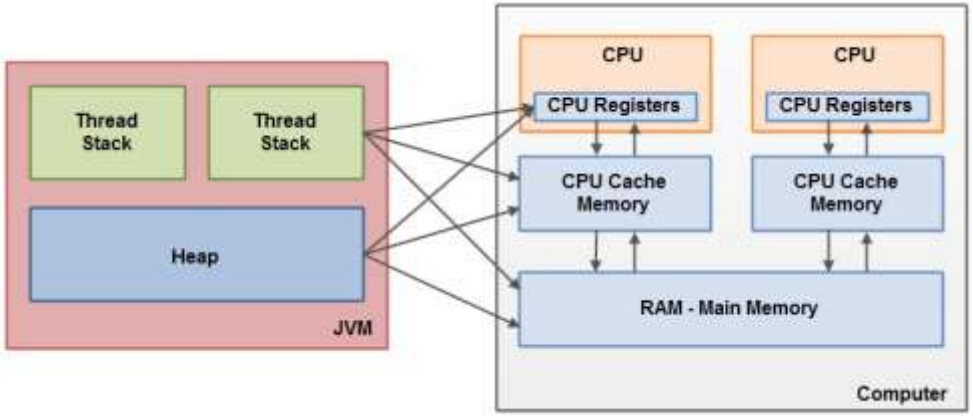
- **De acuerdo a su organización** (Principio de adyacencia) de los elementos se clasifican en:
  - Lineales (Pilas, Listas, Listas, Tablas Hash, Conjuntos, etc.)
  - No lineales (Arboles y Grafos)





- De acuerdo al procesamiento por parte de la computadora
  - **Primitivas** (Procesamiento natural para el CPU o lenguaje de bajo nivel)
  - **No primitivas** (Creadas por el usuario)





- De acuerdo al número de elementos que contiene o contendrá
  - **Dinámicas** (Número de elementos dinámico)
  - **Estáticas** (Número de elementos definido previamente)