

# Especificación de TAD

---

## Lección 2

# Definición de TAD

- Un **Tipo Abstracto de Datos** es un conjunto de valores y de operaciones **definidos mediante una especificación independiente de cualquier representación.**

**TAD = valores + operaciones**

# Esquema

- Características generales de la especificación de un TAD
- Especificación Algebraica
- Especificación no formal

# Programación con TAD: Especificación

## Pasos en la programación con TAD:

1. Especificación del tipo
2. Implementación del tipo
3. Uso del tipo

## 1. Especificación del TAD:

- Establecer la **interfaz** con el usuario del tipo (“lo que necesita saber el usuario-programador”)



Decir qué es, sin decir nada sobre cómo se podría implementar → “**Contrato público**”

- Decidir la lista de operaciones ofrecidas, y especificarlas:
  - para cada operación describir: información que recibe (entrada), información que genera (salida), y el comportamiento o efecto de la operación

# Especificación de TAD

- La especificación de un TAD consiste en establecer las **propiedades** que lo definen.
- Una especificación **útil, debe** ser:
  - Concisa (clara y breve, decir lo imprescindible)
  - General (adaptable a diferentes contextos)
  - Legible (para todos, y que todos entiendan lo mismo)
  - Precisa (no ambigua, evitar problemas de interpretación)



***La especificación define un único tipo, es decir, define totalmente su comportamiento***

**Nota:** Puede ocurrir que dos especificaciones escritas de forma aparentemente distinta, especifiquen exactamente lo mismo, y por lo tanto sean *equivalentes* definiendo exactamente el mismo tipo, con idénticos: conjunto de valores, operaciones y comportamiento.

# Especificación de TAD

→ La especificación de un TAD servirá como un:  
“*contrato público*”

- Alternativas (para “*redactarlo*”):
  - Especificaciones formales → especificación algebraica
    - Precisa, breve, y sin ambigüedades
    - Sintaxis específica para su escritura
    - Verificación formal de TAD
  - Especificaciones no formales
    - Generalmente expresada en *lenguaje natural*
    - También deberá ser precisa, general, legible, no ambigua, única, y definir totalmente su comportamiento

---

**Nota:** Usaremos “formal” o “no formal” en el sentido matemático.

# Especificación Algebraica

- Técnica formal para especificar los TAD
- Objetivo:
  - Definir un tipo de datos sin ambigüedades (conjunto de valores + operaciones permitidas)
- Ventajas:
  - Unanimidad de la interpretación
  - **Definir tipos independientemente de cualquier posible representación**
    - No se pueden usar conceptos como: actualización de un dato en memoria, direcciones de memoria, copias, etc...
  - Verificación formal de los TAD
  - Generación automática de código a partir de la especificación algebraica

# Ejemplo de especificación algebraica

Una  
especificación  
tendrá una parte  
sintáctica...

**espec** misBooleanos

**género** **booleano** ← *El nombre del nuevo tipo*

**operaciones**

*Las operaciones*

**verdad**:  $\rightarrow$  booleano

**falso**:  $\rightarrow$  booleano

$\neg$ \_: booleano  $\rightarrow$  booleano

$\_ \wedge \_$ ,  $\_ \vee \_$ : booleano booleano  $\rightarrow$  booleano

A cada operación se le da un NOMBRE, que puede ser un *identificador* o un *símbolo*, y un perfil (dominio  $\rightarrow$  rango)

Nos indican cómo pueden escribirse términos, bien formados, utilizando dichas operaciones: *Identificador*(argumentos)

*símbolo* argumento

arg1 *símbolo* arg2

**Los identificadores o símbolos elegidos NO les dotan de significado.**

# Ejemplo de especificación algebraica

... y una parte  
semántica

**ecuaciones**  $b$ : booleano;

$$\neg \text{verdad} = \text{falso}$$

$$\neg \text{falso} = \text{verdad}$$

$$b \vee \text{verdad} = \text{verdad}$$

$$b \vee \text{falso} = b$$

$$b \wedge \text{verdad} = b$$

$$b \wedge \text{falso} = \text{falso}$$

*Las propiedades*

*Término = Término*

## **fespec**

- ECUACIONES:
  - Variables representando  $\forall$  término bien formado del género de la variable
  - Se permite el uso de paréntesis para controlar y aclarar el orden de evaluación de las operaciones

# Sintaxis Especificación Algebraica

Una especificación algebraica consta de:

- SIGNATURA, define: → *parte sintáctica*
  - GENEROS: nombres de los nuevos tipos
  - Nombres de las OPERACIONES
  - PERFILES de las operaciones
    - Dominio o aridad
    - Rango o coaridad
- Conjunto de ECUACIONES → *parte semántica*
  - **Cada término** bien formado, que puede escribirse utilizando las operaciones especificadas, **representa un valor distinto** dentro del conjunto de valores del nuevo tipo de dato, salvo que haya ecuaciones que los igualen

# Sintaxis Especificación Algebraica

**espec** boolnat

**géneros** booleano, natural

**operaciones**

verdad , falso:  $\rightarrow$  booleano

$\neg$ \_: booleano  $\rightarrow$  booleano

$\_ \wedge \_$  ,  $\_ \vee \_$ : booleano booleano  $\rightarrow$  booleano

0:  $\rightarrow$  natural

suc: natural  $\rightarrow$  natural

$\_ + \_$  ,  $\_ * \_$ : natural natural  $\rightarrow$  natural

$\_ \leq \_$  ,  $\_ > \_$ : natural natural  $\rightarrow$  booleano

**fespec**

*¡Esta especificación  
no tiene ecuaciones!*

- **Cada término** bien formado, que puede escribirse utilizando las operaciones especificadas, **representa un valor distinto** dentro del conjunto de valores del nuevo tipo de dato.

# Sintaxis Especificación Algebraica

- Definición de los **valores del género**:
  - están representados por los **términos** que se puedan construir (escribir con sintaxis correcta) utilizando únicamente aquello que se define en la especificación
  - el uso correcto de una operación representa un término del género resultado de la operación
  - todos los términos distintos** (no escritos exactamente igual) **representan valores diferentes**

Ejemplo: dada la especificación anterior...

**espec** boolnat

**géneros** booleano, natural

**operaciones**

verdad , falso:  $\rightarrow$  booleano

$\neg$ : booleano  $\rightarrow$  booleano

$\_ \wedge \_$  ,  $\_ \vee \_$ : booleano booleano  $\rightarrow$  booleano

0:  $\rightarrow$  natural

suc: natural  $\rightarrow$  natural

$\_ + \_$  ,  $\_ * \_$ : natural natural  $\rightarrow$  natural

$\_ \leq \_$  ,  $\_ > \_$ : natural natural  $\rightarrow$  booleano

**fespec**

- Son *valores distintos* del género *booleano*:

verdad

falso

$\neg$ verdad

verdad  $\wedge$  falso

...

- Son *valores distintos* del género *natural*:

0

suc(0)

suc(suc(0))

0+0

0+suc(0)

...

# Semántica Especificación Algebraica

**espec** misNaturales

**género** natural

**operaciones**

0:  $\rightarrow$  natural

suc: natural  $\rightarrow$  natural

$\_+ \_$ : natural natural  $\rightarrow$  natural

**ecuaciones**  $x, y$ : natural

$x+0 = x$

$x+\text{suc}(y) = \text{suc}(x+y)$

**fespec**

→ Las ecuaciones son la parte semántica: lo que nos permite definir qué términos distintos representan el mismo valor del género:

0	y	0+0	...
0+suc(0)	y	suc (0+0)	y suc(0)
...			

# Sintaxis Especificación Algebraica

- OPERACIONES:
  - Clasificación en operaciones:  
generadoras, modificadoras, observadoras
  - Notación funcional:
    - Toma como parámetros 0 ó N valores (Dominio o aridad)
    - Produce **un** solo valor **resultado** (rango o coaridad)
  - Las operaciones 0-arias se denominan **CONSTANTES**, de su tipo resultado
    - Sus nombres se utilizan como los literales de los valores que representan
  - Una operación puede tener situaciones indeseadas o de **error**:
    - Cuando existen casos para los cuales no existe un valor válido que pueda representar el resultado de la operación
    - Hacen que la operación sea una **operación parcial**, y deben indicarse en la especificación

# Ejemplo de fechas: Especificación algebraica

**espec** fechas

**usa** enteros, booleanos

**género** fecha

**operaciones**

**parcial** crear: entero entero entero  $\rightarrow$  fecha

día: fecha  $\rightarrow$  entero

mes: fecha  $\rightarrow$  entero

año: fecha  $\rightarrow$  entero

iguales: fecha fecha  $\rightarrow$  booleano

anterior: fecha fecha  $\rightarrow$  booleano

posterior: fecha fecha  $\rightarrow$  booleano

**dominios de definición** d,m,a: entero;

crear(d,m,a) está definida si y solo si  $1 \leq d \leq 31$ ,  $1 \leq m \leq 12$  y  $1583 \leq a$ , y además d, m y a componen una fecha válida, es decir el valor de día d es adecuado para el mes m y el año a, según las reglas del calendario gregoriano

...

No serán las especificaciones de los sencillos ejemplos anteriores. A PARTIR DE AHORA:  
Para los tipos básicos (entero, natural, real, booleano) asumiremos que corresponden a especificaciones completas de los que conocemos de matemáticas, con todas sus propiedades. También asumiremos que tenemos definidos los tipos: carácter y cadena, con sus operaciones y propiedades habituales.

# Ejemplo de fechas: Especificación algebraica

...

**ecuaciones**  $d, d1, d2, m, m1, m2, a, a1, a2$ : entero;  $f1, f2$ : fecha

$\text{dia}(\text{crear}(d, m, a)) = d$

$\text{mes}(\text{crear}(d, m, a)) = m$

$\text{año}(\text{crear}(d, m, a)) = a$

$\text{iguales}(\text{crear}(d1, m1, a1), \text{crear}(d2, m2, a2)) =$   
 $= (d1 = d2) \wedge (m1 = m2) \wedge (a1 = a2)$

$\text{anterior}(\text{crear}(d1, m1, a1), \text{crear}(d2, m2, a2)) =$   
 $= (a1 < a2) \vee ((a1 = a2) \wedge (m1 < m2))$   
 $\vee ((a1 = a2) \wedge (m1 = m2) \wedge (d1 < d2))$

$\text{posterior}(f1, f2) = \neg (\text{iguales}(f1, f2) \vee \text{anterior}(f1, f2))$

**fespec**

# Verificación formal de TAD

La *especificación formal* de TAD facilita la

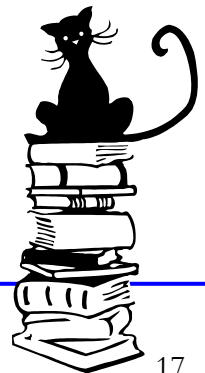
**verificación formal:**  == *razonar formalmente la corrección*

- Se puede demostrar que la **especificación** es correcta
- Se puede demostrar la corrección de una **implementación** a partir de su especificación
  - la implementación es fiel a la especificación
- Por tanto, se puede demostrar la corrección de los programas que hacen **uso** del TAD
- Se pueden construir herramientas para generar automáticamente un *prototipo* del código de implementación, a partir de la especificación del TAD

# Bibliografía

Para saber más sobre las especificaciones algebraicas vistas:

- Lecciones 3 a 6 del libro “Campos Laclaustra, J.: *Estructuras de Datos y Algoritmos*, *Prensas Universitarias de Zaragoza*, *Colección Textos Docentes*, 1995 “  
→ **Anexo 4** del libro de apuntes “Campos Laclaustra, J.: *Apuntes de Estructuras de Datos y Algoritmos*, **segunda edición**, versión 4, 2022  
([https://webdiis.unizar.es/asignaturas/EDA/apuntes\\_EDA.pdf](https://webdiis.unizar.es/asignaturas/EDA/apuntes_EDA.pdf))

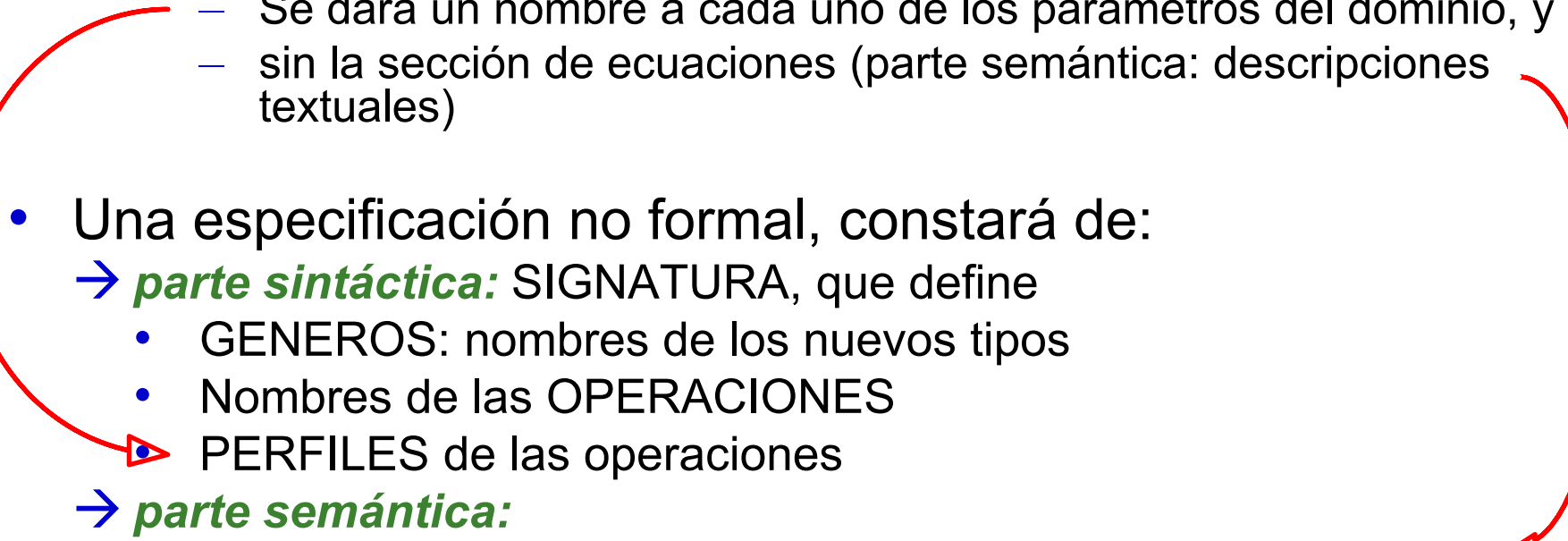


# Especificación no formal de TAD

- Con los mismos objetivos que las especificaciones formales de los TAD:
  - Especificación del TAD debe:
    - Establecer la **interfaz** con el usuario del tipo (“lo que necesita saber el usuario-diseñador/programador”)  
***Decir qué es, sin decir nada sobre cómo se puede implementar***
  - Especificaciones no formales en esta asignatura:
    - *Deberán describir el **conjunto de valores y operaciones** del TAD, y establecer todas las **propiedades** que lo definen, de forma **independiente de cualquier posible representación o implementación del TAD***
- **Especificaciones no formales de TAD** → “contrato público”
  - Generalmente expresadas en lenguaje natural
  - También **deben ser precisas, concisas, generales, legibles, y definir totalmente el comportamiento del TAD y sus operaciones**

¡DIFÍCIL!

# Especificación no formal de TAD

- Especificaciones no formales en esta asignatura:
    - **Sintaxis** similar a la vista para las especificaciones algebraicas:
      - Notación funcional:
        - Toma como parámetros 0 o N valores (Dominio o aridad)
        - Produce **un** solo valor **resultado** (rango o coaridad)
      - Pero con diferencias:
        - Se dará un nombre a cada uno de los parámetros del dominio, y
        - sin la sección de ecuaciones (parte semántica: descripciones textuales)
  - Una especificación no formal, constará de:
    - **parte sintáctica**: SIGNATURA, que define
      - GENEROS: nombres de los nuevos tipos
      - Nombres de las OPERACIONES
      - PERFILES de las operaciones
    - **parte semántica**:
      - **Descripciones textuales** escritas en lenguaje natural pero de forma concisa, general, legible, y no ambigua
- 

# Especificación no formal de TAD

## parte sintáctica:

**espec** nombreEspecificación

[ **usa** especificación1, especificación2, ... ]

...{aquí una parte de la especificación que se verá más adelante}...

**género** nombreGénero1, ...

**operaciones**

[**parcial**] nombreOperación : [ dominio ]  $\rightarrow$  rango

[**parcial**] nombreOperación \_ : géneroParam **nombreParam**  $\rightarrow$  rango

[**parcial**] \_ nombreOp \_ : genParam1 **nomParam1** , genParam2 **nomParam2**  $\rightarrow$  rango

...

**fespec**

- Donde:
- [ ] Significa que lo que está entre los corchetes es opcional, puede aparecer o no
- **Dominio** es una lista de elementos separados por ‘,’ y donde **cada elemento** describe un parámetro de la operación, indicando el **género y nombre del parámetro**
- **Rango** es el nombre de **un género** (notación funcional)
- El símbolo ‘\_’ indica la posición de los argumentos respecto al nombre de la operación. Se utiliza para indicar operaciones con notación prefija sin paréntesis o con notación infija (ejemplo:  $\neg\_ , \_ \leq \_$  )

Disponible en el material de clase:  
documento resumen (chuleta) de la notación no formal que se utilizará en la asignatura para especificar los TAD

# Especificación no formal de TAD

## parte semántica:

**¡cuidado!**

- **No debe haber detalles de posibles implementaciones**
  - *La especificación del TAD es independiente de cualquier posible representación o implementación del TAD*
- **Descripciones textuales** escritas en lenguaje natural pero de forma concisa, general, legible, y no ambigua:
  - A. DESCRIPCION del TAD:**
    - describiendo el **conjunto de valores** que representa el TAD y las **propiedades** que lo definen
  - B. Para CADA OPERACIÓN:** junto a su perfil debe describirse completamente:
    1. Información de entrada y los prerequisites que deban cumplirse para usar la operación (*precondición*)
    2. Comportamiento o efecto de la operación al aplicarse sobre las entradas e indicando qué resultado se genera (*poscondición*)
    3. **Situaciones indeseadas o de error:**
      - Cuando existen casos para los cuales no existe un valor válido que pueda representar el resultado de la operación
      - Hacen que la operación sea una **operación parcial**, y deben indicarse en la especificación

# Especificación no formal de TAD

## partes sintáctica y semántica:

**espec** nombreEspecificación

[ **usa** especificación1, especificación2, ... ]

...{aquí una parte de la especificación que se verá más adelante}...

**género** nombreGénero1, ... **{descripción del TAD}**

**operaciones**

[**parcial**] nombreOperación : [ dominio ]  $\rightarrow$  rango

{**Descripción del dominio y el rango de la operación.**

[**Parcial: descripción de las situaciones que hacen la operación parcial**]}

...

[**parcial**] nombreOperación \_ : géneroParam nombreParam  $\rightarrow$  rango

{**Descripción del dominio y el rango de la operación.**

[**Parcial: descripción de las situaciones que hacen la operación parcial**]}

...

[**parcial**] \_ nombreOperación \_ :

géneroParam1 nombreParam1, géneroParam2 nombreParam2  $\rightarrow$  rango

{**Descripción del dominio y el rango de la operación.**

[**Parcial: descripción de las situaciones que hacen la operación parcial**]}

...

**fespec**

Disponible en el  
material de clase:

documento resumen  
(chuleta) de la notación  
no formal que se  
utilizará en la  
asignatura para  
especificar los TAD

# Ejemplo de fechas: Especificación no formal

**espec** fechas

**usa** enteros, booleanos

Para los tipos básicos (entero, natural, real, booleano) asumiremos que corresponden a especificaciones completas de los que conocemos de matemáticas, con todas sus propiedades. También asumiremos que tenemos definidos los tipos: carácter y cadena, con sus operaciones y propiedades habituales.

**género** fecha

*{DESCRIPCION del TAD: Los valores del TAD fechas representan fechas válidas según las reglas del calendario gregoriano (adoptado en 1583)}*

## Operaciones

**parcial** crear: entero d, entero m, entero a → fecha

*{Dados 3 valores enteros: d, m y a, siendo  $1 \leq d \leq 31$ ,  $1 \leq m \leq 12$  y  $1583 \leq a$ , se obtiene una fecha compuesta con los tres valores dados usados como día, mes y año respectivamente.*

**Parcial:** La operación no está definida (se producirá una situación de error) si los valores d, m y a no pueden formar un valor de fecha válido según el calendario gregoriano (fechaInválida)}

Recomendable darle un nombre a cada caso o situación de error, especialmente si hay varios o si el mismo caso se produce también en otras operaciones

...

# Ejemplo de fechas: Especificación no formal

...

día: fecha  $f \rightarrow$  entero

*{Dada una fecha  $f$ , se obtiene el entero que corresponde al día en la fecha  $f$ }*

mes: fecha  $f \rightarrow$  entero

*{Dada una fecha  $f$ , se obtiene el entero que corresponde al mes en la fecha  $f$ }*

año: fecha  $f \rightarrow$  entero

*{Dada una fecha  $f$ , se obtiene el entero que corresponde al año en la fecha  $f$ }*

iguales: fecha  $f_1$ , fecha  $f_2 \rightarrow$  booleano

*{Dadas dos fechas  $f_1$  y  $f_2$ , se obtiene un booleano con valor verdad si y solo si la fecha  $f_1$  es igual que la fecha  $f_2$ , es decir, corresponden al mismo día, mes y año.}*

anterior: fecha  $f_1$ , fecha  $f_2 \rightarrow$  booleano

*{Dadas dos fechas  $f_1$  y  $f_2$ , se obtiene un booleano con valor verdad si y solo si la fecha  $f_1$  es cronológicamente anterior a la fecha  $f_2$ .}*

posterior: fecha  $f_1$ , fecha  $f_2 \rightarrow$  booleano

*{Dadas dos fechas  $f_1$  y  $f_2$ , se obtiene un booleano con valor verdad si y solo si la fecha  $f_1$  es cronológicamente posterior a la fecha  $f_2$ .}*

# Ejemplo de fechas: Especificación no formal

- Sobre las **operaciones parciales** y los **prerrequisitos**:
  - Los usuarios de la especificación deberán asegurarse de que se cumplan los **prerrequisitos** antes de utilizar la operación
  - **En la especificación se deben indicar las situaciones de error** como información para el usuario del TAD (incluidos los usuarios-programadores del TAD)
- a) **parcial** crear: entero d, entero m, entero a  $\rightarrow$  fecha  
*{(precondición) Dados 3 valores enteros d, m y a, siendo  $1 \leq d \leq 31$  y  $1 \leq m \leq 12$ .*  
...  
*Parcial: la operación no está definida si los valores d, m y a no pueden formar un valor de fecha válido según el calendario gregoriano (fechaInválida) }*
- b) **parcial** crear: entero d, entero m, entero a  $\rightarrow$  fecha  
*{(precondición) Dados 3 valores enteros d, m y a.*  
...  
*Parcial : la operación no está definida si no se cumple que  $1 \leq d \leq 31$  y  $1 \leq m \leq 12$  , o si a pesar de estar en esos rangos los valores d, m y a no pueden formar un valor de fecha válido según el calendario gregoriano (fechaInválida) }*
- Toda implementación debe ser **robusta**:
  - se protegerá frente a valores inconsistentes o que no cumplan los prerrequisitos, y
  - se protegerá de los casos señalados que hacen la operación parcial

**Ejercicio:** Diseñar un programa que lea una secuencia de enteros de un fichero, mayores que 0, y al acabar escriba en pantalla cada entero leído del fichero junto con su frecuencia de aparición (total de apariciones del número en el fichero), siguiendo el orden de frecuencias decrecientes.

