Instituto Politécnico Nacional
La Técnica al Servicio de la Patria

Unidad Profesional Interdisciplinaria de Ingeniería Campus Tlaxcala UPIIT

Algoritmos y Estructuras de Datos

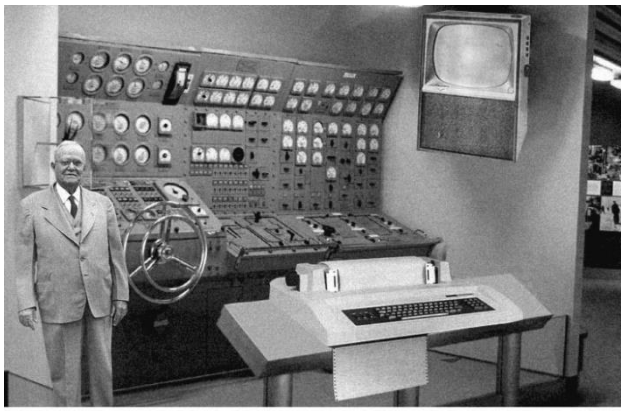
Esaú Eliezer Escobar Juárez

Ingeniería en Inteligencia Artificial (IIA)

Introducción a la abstracción de datos

Introducción a la abstracción de datos

- ¿Qué es abstracción?
  - La abstracción consiste en aislar un elemento de su contexto o del resto de los elementos que lo acompañan.



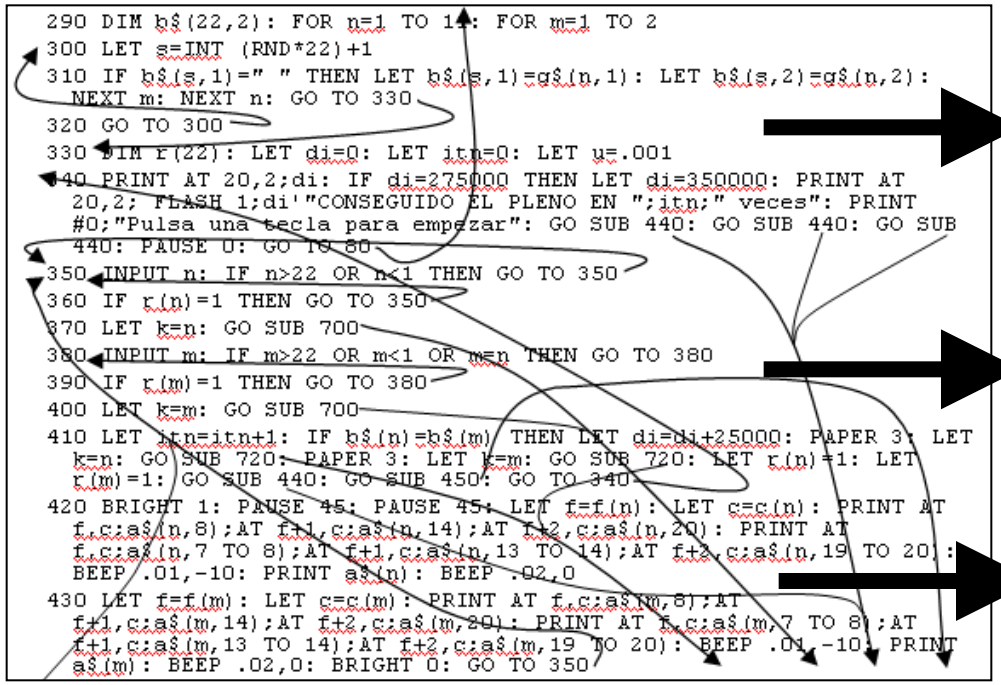
Scientists from the RAND Corporation have created this model to illustrate how a "home computer" could look like in the year 2004. However the needed technology will not be economically feasible for the average home. Also the scientists readily admit that the computer will require not yet invented technology to actually work, but 50 years from now scientific progress is expected to solve these problems. With teletype interface and the Fortran language, the computer will be easy to use and only

Tipos de datos abstractos

- Este concepto aparece a mediados de la década de los 70's

- Tipo abstracto de dato (TDA)
  - conjunto de valores que lo caracteriza.
  - operaciones que sobre él se pueden aplicar.
  - propiedades que determinan inequívocamente su comportamiento.

Introducción a la abstracción de datos



```
290 DIM b$(22,2): FOR n=1 TO 14: FOR m=1 TO 2
300 LET s=INT (RND*22)+1
310 IF b$(s,1)=" " THEN LET b$(s,1)=g$(n,1): LET b$(s,2)=g$(n,2):
  NEXT m: NEXT n: GO TO 330
320 GO TO 300
330 DIM r(22): LET di=0: LET itn=0: LET u=.001
340 PRINT AT 20,2;di: IF di=275000 THEN LET di=350000: PRINT AT
  20,2; FLASH 1;di'"CONSEGUIDO EL PLENO EN ";itn;" veces": PRINT
  #0;"Pulsa una tecla para empezar": GO SUB 440: GO SUB 440: GO SUB
  440: PAUSE 0: GO TO 80
350 INPUT n: IF n>22 OR n<1 THEN GO TO 350
360 IF r(n)=1 THEN GO TO 350
370 LET k=n: GO SUB 700
380 INPUT m: IF m>22 OR m<1 OR m=n THEN GO TO 380
390 IF r(m)=1 THEN GO TO 380
400 LET k=m: GO SUB 700
410 LET itn=itn+1: IF b$(n)=b$(m) THEN LET di=di+25000: PAPER 3: LET
  k=n: GO SUB 720: PAPER 3: LET k=m: GO SUB 720: LET r(n)=1: LET
  r(m)=1: GO SUB 440: GO SUB 450: GO TO 340
420 BRIGHT 1: PAUSE 45: PAUSE 45: LET f=f(n): LET c=c(n): PRINT AT
  f,c;a$(n,8);AT f+1,c;a$(n,14);AT f+2,c;a$(n,20): PRINT AT
  f,c;a$(n,7 TO 8);AT f+1,c;a$(n,13 TO 14);AT f+2,c;a$(n,19 TO 20):
  BEEP .01,-10: PRINT a$(n): BEEP .02,0
430 LET f=f(m): LET c=c(m): PRINT AT f,c;a$(m,8);AT
  f+1,c;a$(m,14);AT f+2,c;a$(m,20): PRINT AT f,c;a$(m,7 TO 8);AT
  f+1,c;a$(m,13 TO 14);AT f+2,c;a$(m,19 TO 20): BEEP .01,-10: PRINT
  a$(m): BEEP .02,0: BRIGHT 0: GO TO 350
```

procedure ordenar(array a, array b);

type persona;

class pila;

Abstraer: Eliminar lo irrelevante y quedarnos con lo realmente importante.

¿Qué es lo importante?

Introducción a la abstracción de datos

TIPOS DE ABSTRACCIONES

- Abstracciones funcionales → Rutinas, funciones, procedimientos
- Abstracciones de datos → Tipos Abstractos de Datos (TAD)
- Abstracciones de iteradores → Iteradores

Introducción a la abstracción de datos

TIPO ABSTRACTO DE DATOS:

Dominio abstracto de valores junto con las operaciones aplicables sobre el mismo.

TIPO DE DATOS:

Conjunto de valores que puede tomar una variable, un parámetro o una expresión.

ESTRUCTURA DE DATOS:

Disposición en memoria de los datos necesarios para almacenar valores de un tipo.

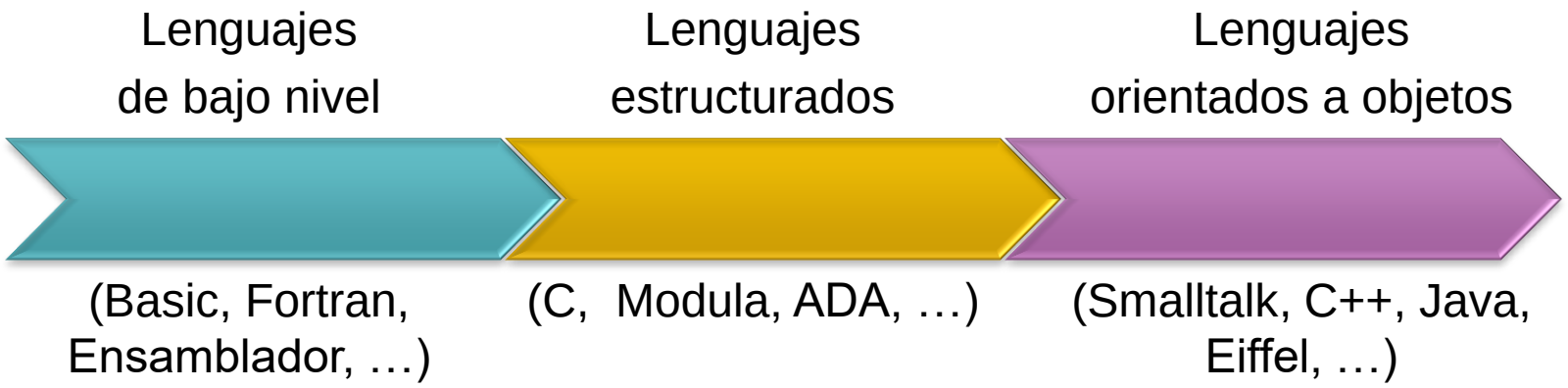
Introducción a la abstracción de datos

Ejemplos

- **TAD**: Enteros
- **Tipo de datos**: Tipo **int** de C/C++
- **Estructura de datos**: Representación mediante enteros de 16 bits, 32 bits, listas de dígitos (enteros largos), etc.

Introducción a la abstracción de datos

- La evolución de los lenguajes de programación tiende a introducir cada vez más abstracciones.



Soporte de TAD:

Lenguajes estructurados (tipos definidos por el usuario): Los datos y las operaciones van aparte.

Lenguajes orientados a objetos (clases): Los datos y las operaciones constituyen una unidad, el concepto de clase.

C

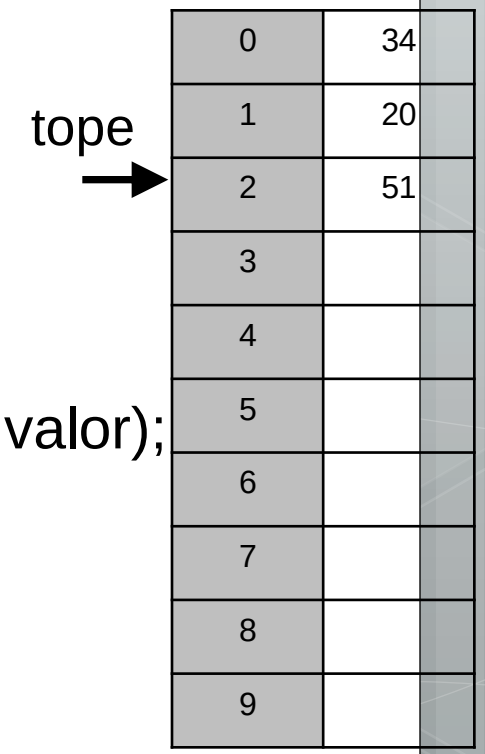
```
struct Pila {
    int tope;
    int datos[10];
};

void push (Pila *p, int valor);
void pop (Pila *p);
int top (Pila p);
```

C++

```
class Pila {
  private:
    int tope;
    int datos[10];
  public:
    void push (int valor);
    void pop ( );
    int top ( );
};
```

Pila



C

```
Pila p1, p2;
int i;

push(&p1, 34);
push(&p1, 20);
push(&p1, 51);
pop(&p1);
i= top(p1);
p1.tope= 243;
i= top(p1);
...
```

Error de ejecución: se sale del array datos

C++

```
Pila p1, p2;
int i;

p1.push(34);
p1.push(20);
p1.push(51);
p1.pop();
i= p1.top();
p1.tope= 243;
i= p1.top();
...
```

Error de compilación: tope es privado

Pila

datos

tope

0	34
1	20
2	51
3	
4	
5	
6	
7	

Tipo abstracto de datos

- ¿Por qué abstracto?
  - Responde al hecho de que los valores de un tipo pueden ser manipulados mediante sus operaciones si se saben las propiedades que éstas cumplen, sin que sea necesario ningún conocimiento acerca de su implementación en la máquina.

Propiedades de los TDA

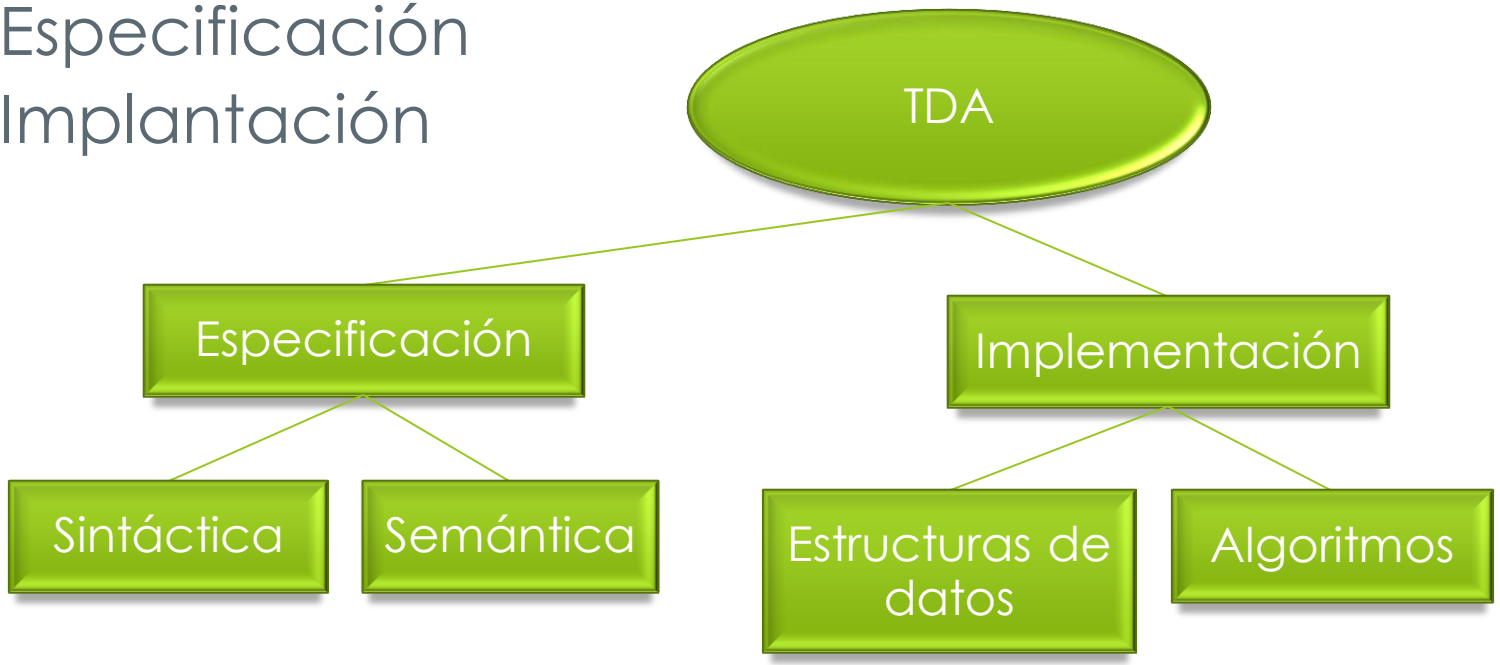
- Con abstracción, creamos una entidad bien-definida, la cual puede ser adecuadamente manejada.
- Estas entidades definen la estructura de datos de un conjunto de ítems o elementos.
  - Por ejemplo, cada empleado administrado tiene un nombre, fecha de nacimiento, número de seguridad social …
- La **estructura de datos** solamente puede ser accedida con operaciones bien definidas.
  - Este conjunto de operaciones se llama interface y es exportada por la entidad.
- Una **entidad** con las propiedades justamente descritas se llama tipo de dato abstracto (TDA).

Definición de un TDA

- **TDA** se caracteriza por las siguientes propiedades:
  1. Exporta un tipo.
  2. Exporta un conjunto de operaciones. Este conjunto es llamado interface.
  3. Las Operaciones de la interface son el único mecanismo de acceso a la estructura de datos del tipo.
  4. Axiomas y precondiciones definen el dominio de la aplicación del tipo.

Objetivo de un TDA

- Separar el uso del tipo de dato, de su implementación.
- Para conseguir este objetivo, la definición de un TDA se divide en dos partes:
  - Especificación
  - Implantación



Especificación Sintáctica

- Qué hace? Especificación de las entidades y sus propiedades (interface).
- Definir el nombre de las entidades abstractas.
- Definir el nombre de las operaciones indicando el dominio (argumentos) y el co-dominio o rango (los valores de retorno).

Especificación Semántica

- ¿Cómo lo hace? Descripción de la representación del objeto (estructuras de los datos) y desarrollo de las operaciones.
- Definir el significado de cada operación usando los símbolos definidos en la especificación sintáctica.
- La especificación puede ser de dos tipos:
  - Informal, a través del lenguaje natural.
  - Formal, rigurosa y fundamentada matemáticamente.

Especificación Semántica

- Además:
  - {P} Pre-condición: condiciones que deben cumplirse antes de realizar la operación.
  - {Q} Post-condición: condiciones que se cumplen una vez realizada la operación.

- La notación usual es {P} S {Q}, donde S es la función o procedimiento.

Especificación de un TDA

- La especificación de un TDA consiste en establecer las propiedades que lo definen.

- Para describir un TDA es necesario describir:
  - Los valores que pueden tomar los datos de ese tipo.
  - Todas las operaciones realizables sobre de ellos.

- Una especificación debe poseer 4 propiedades:
  - Ser precisa: Solo dice lo imprescindible.
  - Ser general: Es adaptable a diferentes contextos.
  - Ser legible: Transmite a los usuarios del tipo y al implementador el comportamiento del tipo.
  - No ambigua: Evita dobles interpretaciones.

Ejemplo: TDA Bolsa

- **Definición:** Es una colección no ordenada de elementos con repetición.
- **Tipo:** Bolsa.
- **Sintaxis:**

  CrearBolsa() → Bolsa
  BolsaVacia → Bolsa
  BolsaLlena → Bolsa
  Poner(Bolsa, Objeto) → Bolsa
  EsVacia(Bolsa) → Boolean
  Retirar(Bolsa, Objeto) → Objeto
- **Semántica:** b es Bolsa, e, f son elementos

  CrearBolsa() = BolsaVacia
  EsVacia(CrearBolsa()) = Verdadero
  EsVacia(Poner(CrearBolsa(), e)) = Falso
  Retirar (BolsaVacia, e) = Error
  Retirar (Poner(CrearBolsa(), f), e) = f si f=e
  Poner (BolsaLlena, e) = Error

Ejemplo: TDA Bolsa

- **Definición:** Es una colección no ordenada de elementos con repetición.
- **Tipo:** Bolsa.
- **Operaciones:**

  Función **Construir_Bolsa** () → bolsa

  {postcondición: Devuelve una bolsa vacía}

  Función **Poner** (B: bolsa; e: elemento) → bolsa

  {precondición: La bolsa no esta llena}

  {postcondición: Añade el elemento e a la bolsa}

  Función **EsVacia** (B: bolsa) → boolean

  {postcondición: Devuelve verdadero si la bolsa no tiene elementos, falso en otro caso}

  Función **Retirar** (B: bolsa; e: elemento) → elemento

  {precondición: La bolsa no esta vacía}

  {postcondición: Elimina el elemento e de la bolsa B}

Ejemplo: TDA Fracción

- **Definición:** Una fracción es un par ordenado de enteros siempre y cuando la segunda componente sea distinta de cero.
- **Tipo:** Fracción.
- **Tipo abstracto Racional:** (e(1), e(2)).
- **Operaciones:**

  Función **ConstruirFraccion** (a,b: Entero)→ Fracción
  {precondición: b ≠ 0}
  {postcondición: e(1) = a AND e(2) = b}
  Función **Numerador** (r: Fracción) → entero
  {postcondición: Numerador = e(1)}
  Función **Denominador** (r: Fracción) → entero
  {postcondición: Denomidador = e(2)}
  Función **Multiplicar** (r1, r2: Fracción) → Fracción
  {Inicio multiplicar = contruirFraccion (Numerador(r1) * Numerador(r2), Denominador(r1) * Denominador(r2))
  Fin}

Ejemplo TDA bool

- **Definición:** Tipo booleano
- **Tipo:** bool
- **Sintaxis:**
  - cierto, falso → bool
  - ~ (bool) → bool
  - V(bool, bool) → bool
  - Λ (bool, bool) → bool
- **Semántica:**
  - ~ cierto = falso
  - b V cierto = cierto
  - b V falso = b
  - b Λ cierto = b
  - b Λ falso = falso

Ejercicio

- Vector
- Número complejo
- Triángulo