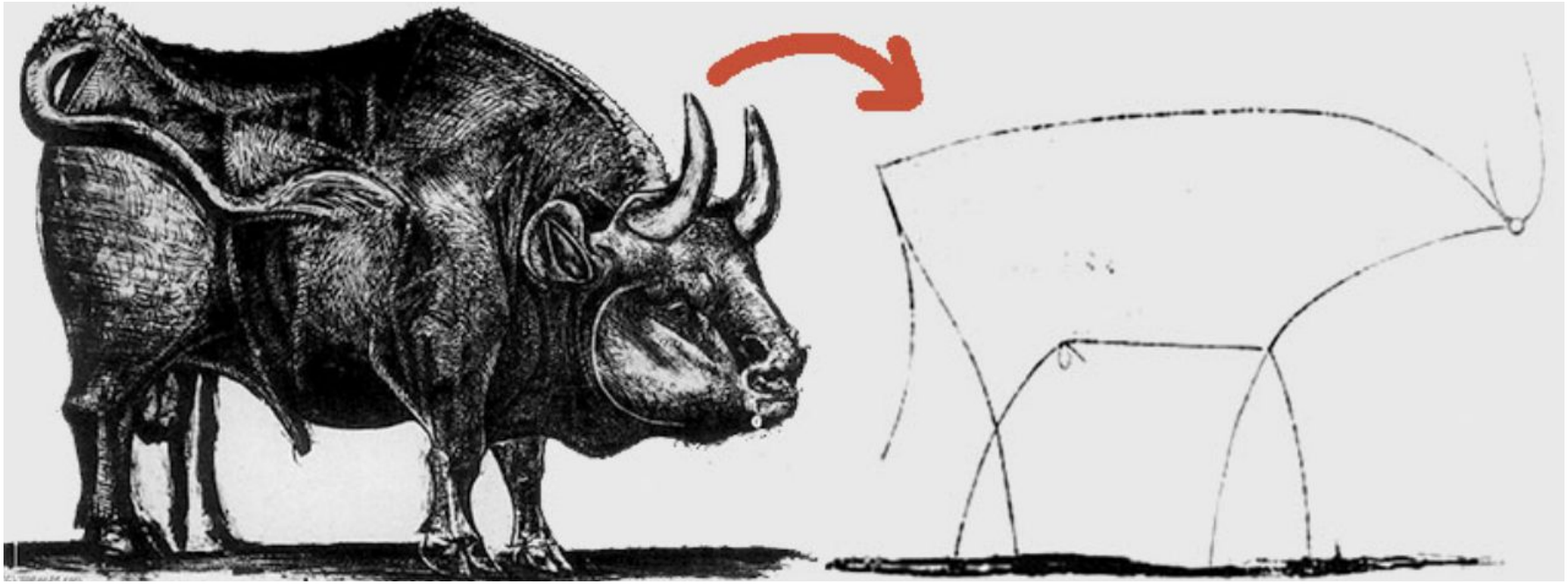
TDA

ABSTRACCIÓN



ABSTRACCIÓN

“La Abstracción es el proceso por el cual se despoja a un problema o cosa de la complejidad que no es relevante para la solución de un problema determinado.”

Siguiente paso de abstracción

En el lenguaje de programación C las herramientas que hacen extensible al lenguaje con nuevas abstracciones son:

- Funciones y procedimientos
- Archivos de encabezado
- Tipos de datos primitivos del lenguaje

Tipo de Dato Abstracto (TDA)

“Define una clase de objetos abstractos los cuales están completamente caracterizados por las operaciones que pueden realizarse sobre esos objetos.”

Tipo de Dato Abstracto (TDA)

“En otras palabras, un tipo de dato abstracto está definido no sólo por una estructura de datos sino también por las operaciones que se define sobre esa estructura, es decir que son ambas cosas juntas.”

Tipo de Dato Abstracto (TDA)

“En otras palabras, un tipo de dato abs[illegible]nido no sólo por una estructura d[illegible]mbién por las operaciones que s[illegible]esa estructura, es decir que son amb[illegible].

STRUCT + FUNCIONES

Primitivas

Estas funciones son un conjunto de operaciones que describen el comportamiento de un TDA y se llaman primitivas.

Una importante implicación del uso de un TDA, es que la mayoría (por no decir todas) de las operaciones que se utilizan en un programa de un determinado tda pertenecen al set de operaciones características o primitivas del mismo.

C VS Python (primer clase)

Los arrays o vectores en C almacenan muchos elementos de un mismo tipo de dato. Tienen un tamaño fijo que se determina al momento de la declaración del vector, y no pueden almacenar más elementos que su tamaño (sí menos).

En python tenemos las listas, la diferencia principal con los vectores de C es que son dinámicas (no hay que andar pidiendo y devolviendo memoria), y no tienen un tamaño fijo.
Además, la lista puede guardar cualquier tipo de dato.

C VS Python (primer clase)

```c
// creación array vacío
int array[5];

// creación array con elementos
int array[5] = {1, 2, 3, 4, 5};

// agregar un elemento
array[0] = 5;

// acceso al primer elemento
array[0];
```

```python
# creación vacío
array = []

# creación con elementos
array = [1, 2, 3, 4, 5]

# agregar un elemento
array.append(5)

# acceso al primer elemento
array[0]
```

Objetivo

Los TDA deben tender a parecerse a un tipo de dato primitivo provisto por el lenguaje de programación. Uno usa variables de tipo entero y no se pregunta como la computadora implementa internamente ese tipo, utiliza sus operaciones características o primitivas como la suma, resta, multiplicación, división entera y resto de la división entera.

Ejemplo: TDA Número Complejo

```c
typedef struct n_complejo n_complejo_t;

// Crea un número complejo con parte real e imaginaria
n_complejo_t* ncomp_crear(double real, double imag);

// Destruye un número complejo liberando la memoria
void ncomp_destruir(n_complejo_t* num);

// Suma dos números complejos
n_complejo_t* ncomp_sumar(const n_complejo_t* a, const n_complejo_t* b);

// Resta dos números complejos
n_complejo_t* ncomp_restar(const n_complejo_t* a, const n_complejo_t* b);

// Multiplica dos números complejos
n_complejo_t* ncomp_multiplicar(const n_complejo_t* a, const n_complejo_t* b);

// Divide dos números complejos
n_complejo_t* ncomp_dividir(const n_complejo_t* a, const n_complejo_t* b);

// Devuelve el conjugado de un número complejo
n_complejo_t* ncomp_conjugado(const n_complejo_t* a);

// Imprime el número complejo en formato a + bi
void ncomp_imprimir(const n_complejo_t* a);
```

Objetivo de TDAs en Fundamentos de Programación

- Qué son
- Cómo es su estructura
- Cómo se usan

Ventajas: Manejan la Abstracción

Son la herramienta perfecta para manejar la abstracción que permite simplificar la realidad mediante el despojo de complejidad que no es inherente al problema en estudio. Es más los tdas permiten crear niveles de abstracción basándose en construir tdas nuevos usando tipos primitivos y otros tdas

Ventajas: Encapsulamiento

Un tda debe exponer la menor cantidad posible de información del cómo está implementado, y debe por ende hacer que el usuario del tda se base en las funciones que el mismo le provee. El implementador de un tda debe poder ocultar la mayor cantidad de detalles sobre la implementación y el diseño del mismo

Ventajas: Localización del Cambio

Cuando existe un error dentro de un programa es mucho más fácil detectarlo pues la utilización de los tdas fuerza la modularización.