Dominando Estruturas de Dados 1

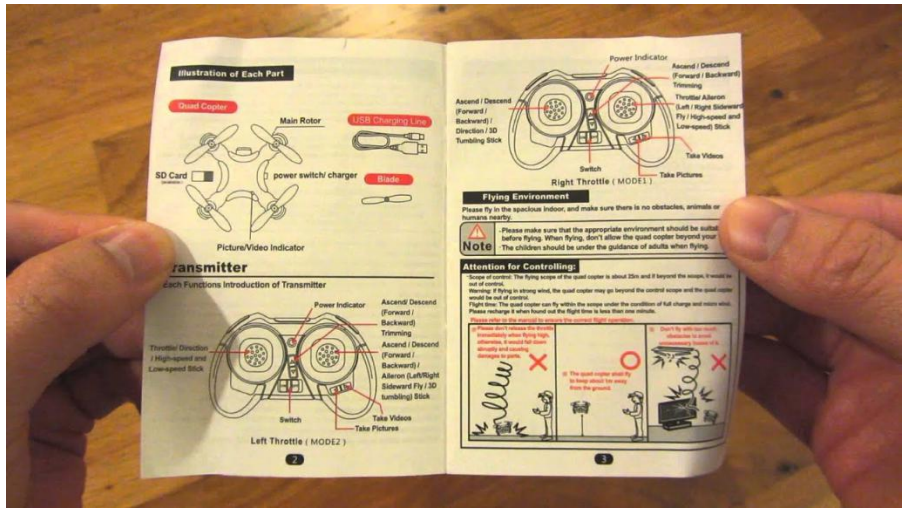
Tipos Abstratos de Dados

Prof. dr. Samuel Martins (Samuka)
@xavecoding @hisamuka

xavecoding

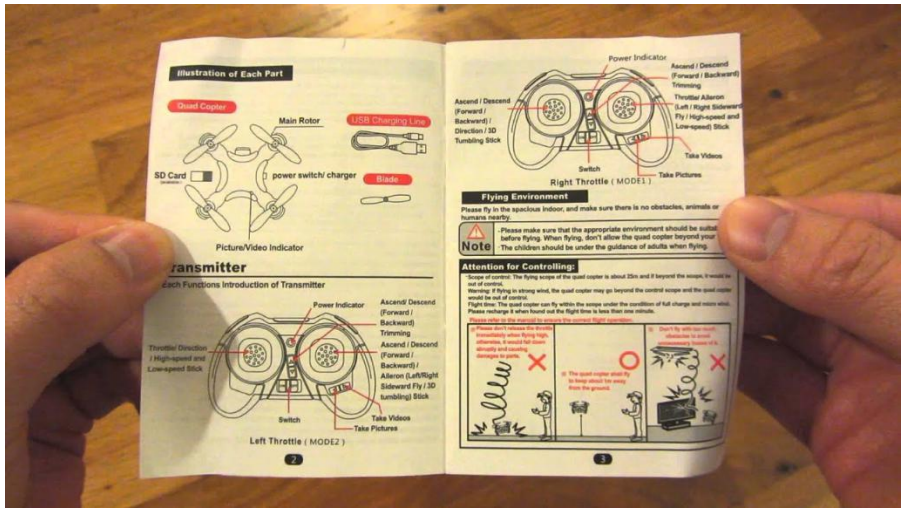
CC BY NC SA

Tipo Abstrato de Dados (TAD)



Tipo Abstrato de Dados (TAD)

O usuário **não está preocupado** como **as funções** do drone e do controle remoto **foram implementadas**. Ele só se interessa em usar tais funções.



A fabricante do drone pode **definir as especificações das funções** que devem estar presentes nos controles remotos. Assim, outras empresas podem **implementá-las**, produzindo diferentes controles remotos para o mesmo drone.

Tipo Abstrato de Dados (TAD)

- Visa **desvincular** o **tipo de dado** (**estrutura de dados** e **operações** que as manipulam) de sua **implementação;**
- Quando definimos um **Tipo Abstrato de Dados (TAD)** estamos preocupados com **o que ele faz** (**especificação**) e **não como ele faz** (**implementação**);
- Ideia parecida com **Encapsulamento** em Orientação a Objetos:
    - Escodemos os dados e detalhes do usuário, fornecendo apenas uma **interface pública** (métodos/operações) para manipulá-los.

Tipo Abstrato de Dados (TAD)

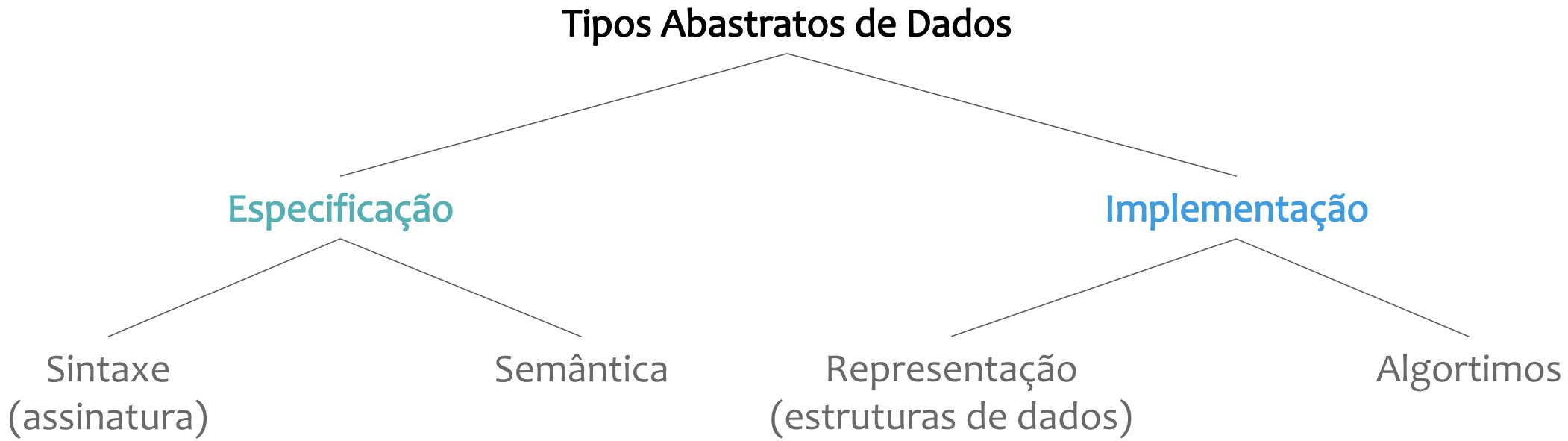
Pense em TADs como **contratos** entre pessoas (ou programas), quem escrevem as **implementações** , e seus clientes.

Tais contratos **devem** incluir todas as **especificações** acordadas:

- o **nome do tipo**, para que o cliente possa referenciá-lo;
- os nomes e **assinaturas** de todas as **operações/funções/algoritmos** primitivas sobre os dados daquele tipo;
- as condiçõoes sob as quais estas operações são aplicáveis;

Os contratos **NÃO DEVEM** incluir:

- informações sobre a atual **representação** do tipo;
- informações sobre **implementações** das operações/funções associadas ao tipo;



PS: Separação de **especificação** e **implementação**: permite o uso do TAD sem conhecer nada sobre a sua implementação

http://www.decom.ufop.br/anascimento/site_media/uploads/bcc202/aula_03_-_tipo_abstrato_de_dados.pdf

Vantagens de TAD

- **Reutilização:**
  - abstração de datalhes da implementação
- **Facilidade de manutenção:**
  - mudanças na implementação do TAD **não afetam** o código fonte dos programas que o utilizam (ocultamento de informação)
- **Corretude:**
  - códigos testados em diferentes contextos.

TAD em Linguagem C

- Separamos a **especificação** do tipo em um **arquivo de cabeçalho (.h)** e sua **implementação** em um arquivo fonte **.c**
  - **seu_tad.h:** **especificação** da TAD
  - **seu_tad.c:** **implementação** da TAD

- Os programas ou outras TADs que utilizam seuTAD devem incluir sua especificação:

#include “seu_tad.h”

Hands on!

Crie um TAD de um vetor de float:

- O vetor tem uma capacidade máxima (número máximo de elementos);
- O vetor informa seu tamanho (quantidade de elementos armazenados atualmente);

Funções

- size(tad vector): retorna o tamanho do vetor (número atual de elementos inseridos)
- capacity(tad vector): retorna a capacidade do vetor (número maáximo de elementos)
- at (tad vector, int index) : retorna o elemento do índice `index` com bound-checked
- get (tad vector, int index): retorna o elemento do índice `index`
- append(tad vector, float val): adiciona o valor `val` no final do vetor. Lança um erro se o vetor estiver cheio.
- set(tad vector, int index, float val): Atribui o valor `val` no índice `index` do vetor de tipo `tad`. Lança um erro se o índice for inválido.
- print(tad vector): Imprime todos os elementos do vetor.

Exercícios

Continue o TAD:

- remove(tad vector, int index)
  - remove o elemento da posição `index`. Move todos os elementos subsequentes para a esquerda após a remoção. Se o índice for inválido, nada acontece.
- erase(tad vector): limpa o vetor, removendo todos seus elementos
- clone(tad vector): retorna uma cópia/clone do vetor

Dominando Estruturas de Dados 1

Tipos Abstratos de Dados

Prof. Samuel Martins (Samuka)
@xavecoding @hisamuka

xavecoding

CC BY NC SA