# Algum artista na sala?

**Se você é o “Salvador Dalí” do Paint, por favor, um passo à frente**

# Recados

# Tipos Abstratos de Dados (TAD)

**Estruturas de Dados 2018/1**
**Prof. Diego Furtado Silva**
Baseado no material do professor Gustavo E. A. P. A. Batista

# TD, TAD e ED

Vamos discutir 3 temas relacionados

- Tipo de dados
- Tipo abstrato de dados
- Estruturas de dados

# Tipo de dados

Quem lembra?

# Tipo de dados

O tipo de dados está diretamente relacionado à linguagem de programação

Define o **conjunto de valores** que pode assumir e as **operações** disponíveis para sua manipulação

Ex: Variável inteira pode assumir valores em $\mathbb{N}$ e pode ser manipulada por meio de operações aritméticas. Variável do tipo string suporta operações específicas, como concatenação e busca por *substring*.

# Tipo de dados

A quantidade de bytes reservado a uma variável e como ela é representada e manipulada em memória é especificada em sua declaração

- **Tipificação forte** vs. tipificação fraca + inferência de tipo

# Novo tipo de dados

É possível criar **novos tipos de dados** a partir dos tipos existentes. Ex:

- Números complexos
- Conjuntos
- Grandes inteiros
- Imagem
- Áudio
- etc

# Novo tipo de dados

Cada novo tipo de dados pode ser **implementado de diversas maneiras**, podendo haver diferenças em termos de **eficiência**

- Bem como na clareza da implementação e outros elementos

Qualquer implementação deve definir **mesmo domínio e não mudar o significado das operações**

- Criar, inserir, somar, etc

# Novo tipo de dados - Conjunto

Armazena elementos de um tipo específico, sem ordem específica e sem repetição

- Operações de um conjunto
  Adicionar, remover, tamanho, verificar pertinência, etc
- Operações entre conjuntos
  União, intersecção, diferença, subconjunto

# Novo tipo de dados - Conjunto

Como podemos implementar um conjunto

- Vetor ordenado
- Vetor não-ordenado
- *Bit array* (vetor de bits)

  Vetores binários para descrever conjuntos de *n* inteiros *1..n* (ou *0..n-1*)

# Novo tipo de dados - Conjunto

Como podemos implementar um conjunto

- Vetor ordenado
- Vetor não-ordenado
- *Bit array* (vetor de bits)

  Vetores binários para descrever conjuntos de *n* inteiros *1..n* (ou *0..n-1*)

**Qual é o custo de inserir, remover e verificar pertinência em cada uma dessas implementações?**

# Melhorando a implementação

Prática comum (*refactoring*), porém:

- Re-trabalho
- Inserção de erros
- Perda da integridade/coesão

# Melhorando a implementação

Como podemos **modificar a implementação** dos tipos de dados com o menor impacto possível para os programas que o usam?

# Melhorando a implementação

Como podemos modificar a implementação dos tipos de dados com o menor impacto possível para os programas que o usam?

Podemos **“esconder” a implementação** do tipo de dados de que o utiliza para implementar um programa?

# Tipo Abstrato de Dados (TAD)

Tipo de dados completamente desvinculado da

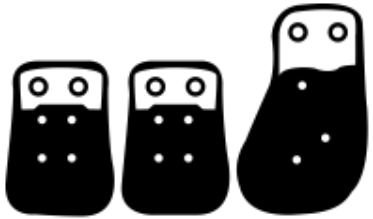
Definido pelo par (*v*,*o*)

- *v*: valores/dados a serem manipulados
- *o*: operações de manipulação de *v*

# Tipo Abstrato de Dados (TAD)

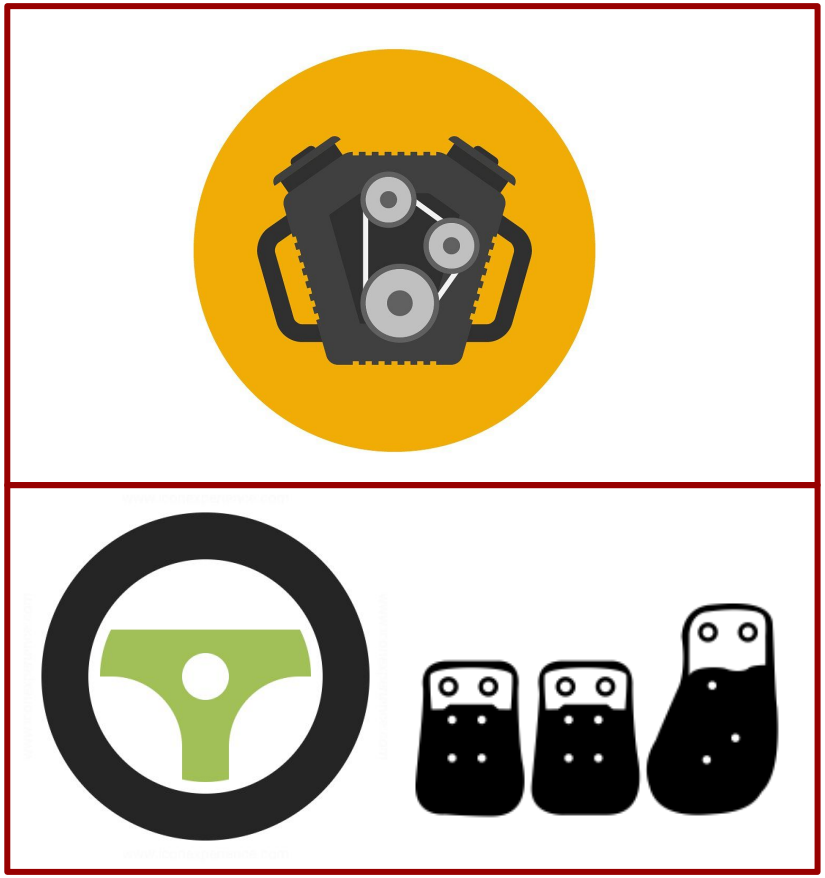
Ao definir um TAD, estamos interessados em saber **o que** ele faz, **não como** ele faz

# TAD Carro



# TAD Carro

Implementação



Interface

# TAD vs. POO

Ideia similar ao **encapsulamento**, relacionado à programação orientada a objetos

- Mas não é a mesma coisa!
- A semelhança é apenas o conceito de **abstração**, ou seja, a separação entre conceito e implementação

# TAD vs. POO

Artigo interessante:

https://www.cs.utexas.edu/~wcook/papers/OOPvsADT/CookOOPvsADT90.pdf

Q&A com alguns *insights* interessantes:

https://www.quora.com/In-OOP-what-is-the-difference-between-an-abstract-data-type-and-a-class-or-object

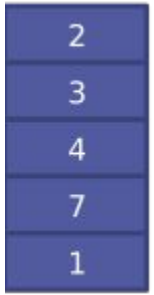
# TAD e Estrutura de Dados

“Uma estrutura de dados é um modo particular de armazenamento e organização de dados em um computador de modo que possam ser usados eficientemente, facilitando sua busca e modificação.”

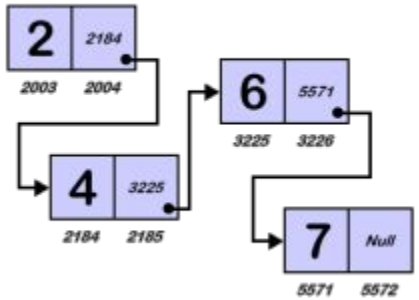
Definição na Wikipedia (em 09/03/2018)

# TAD e Estrutura de Dados

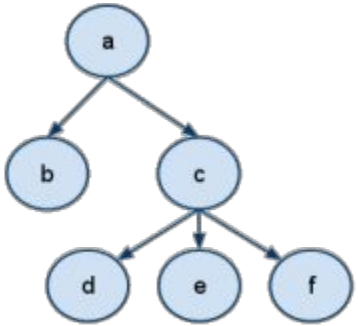
Uma vez que um TAD é definido, escolhe-se a estrutura de dados mais apropriada para representá-lo. Exs:




Arranjo




Lista ligada




Árvore

# Exercício - TAD Conjuntos

Um conjunto é uma coleção de elementos em que  cada elemento ou é um conjunto ou um elemento primitivo chamado de átomo.

Todos os membros são diferentes: nenhum conjunto contém 2 cópias do mesmo elemento.
Exemplo:

- {1,4} - correto
- {1,4,1} - errado

# Exercício - TAD Conjuntos

- Se A e B são conjuntos, então A ∪ B é o conjunto de elementos que são membros de A ou de B ou de ambos
- Se A e B são conjuntos, então A ∩ B é o conjunto de elementos que estão em A e em B
- Se A e B são conjuntos, então A – B é o conjunto de elementos em A que não estão em B

- Exemplo: A = {a,b,c}, B = {b,d}
  A ∪ B = {a,b,c,d}
  A ∩ B = {b}
  A – B = {a,c}

# Exercício - TAD Conjuntos

- Se A e B são conjuntos, então A ∪ B é o conjunto de elementos que são membros de A ou de B ou de ambos
- Se A e B são conjuntos, então A ∩ B é o conjunto de elementos que estão em A e em B
- Se A e B são conjuntos, então A – B é o conjunto de elementos em A que não estão em B

- Exemplo: A = {a,b,c}, B = {b,d}
  A ∪ B = {a,b,c,d}
  A ∩ B = {b}
  A – B = {a,c}

# Exercício - TAD Conjuntos

Vamos usar *bit array* e “implementar” as seguintes operações

- Inicializa(A)
- União(A,B,C)
- Intersecção(A,B,C)
- Diferenca(A,B,C)
- Pertence(x,A)
- Insere(x,A)
- Remove(x,A)

# Exercício - TAD Conjuntos

- Inicializa(A): faz o conjunto vazio ser o valor para a variável conjunto A
- União(A,B,C): toma os argumentos A e B que são conjuntos e retorna A ∪ B à variável C
- Intersecção(A,B,C): toma os argumentos A e B que são conjuntos e retorna A ∩ B à variável C
- Diferença(A,B,C): toma os argumentos A e B que são conjuntos e retorna A - B à variável C
- Pertence(x,A): toma o conjunto A e o objeto x cujo tipo é o tipo do elemento de A e retorna um valor booleano – true se x ∈ A e false caso contrário

# Exercício - TAD Conjuntos

- Insere(x,A): toma o conjunto A e o objeto x cujo tipo é o tipo do elemento de A e faz x um membro de A. O novo valor de A = A $\cup$ {x}. Se x já é um membro de A, então a operação insere não muda A

- Remove(x,A): remove o objeto x, cujo tipo é o tipo do elemento de A, de A. O novo valor de A = A – {x}. Se x não pertence a A então a operação remove não altera A

# Exercício - TAD Conjuntos

Agora, idealize as mesmas operações usando

- Vetor ordenado
- Vetor não-ordenado

Como elas impactam na complexidade das operações?