

## Unidade 2

# Tipos Abstratos de Dados

Qual é a Melhor Maneira para Aumentar o Volume da TV?

### Nossos Objetivos nesta Unidade:

- Entender o conceito de Tipos Abstratos de Dados; entender também porque e como esse conceito deve ser utilizado no desenvolvimento de programas de computador.

### Precisamos Desenvolver um Software: e Agora?

Você e um grupo de amigos foram contratados para desenvolver um software. E agora: qual o primeiro passo? Como vocês dividem o trabalho entre os elementos do grupo? Ficam todos em frente a um único computador, cada um dando seu palpite?

A Figura 2.1 apresenta as principais fases da vida de um software. Inicialmente, na fase de Análise, é preciso determinar **o que** o software precisa fazer, quais problemas deverá resolver, quais informações deverá manipular.

**Análise**

**Projeto**

**Implementação**

**Teste**

**Manutenção**

## Figura 2.1 Ciclo de vida *tradicional* para desenvolvimento de software

Note que na fase de Análise determinamos **o que** o software deverá fazer, mas não definimos ainda **como** o software será desenvolvido. Na fase de Projeto, sim, elaboramos um modelo abstrato (ou seja, um modelo geral, não muito detalhado) de **como** o software será desenvolvido. Na fase de Projeto determinamos, por exemplo, os principais módulos do programa, e com isso podemos dividir o trabalho entre os desenvolvedores, cada um sendo responsável pelo desenvolvimento de um dos módulos.

As outras fase do ciclo de vida do software referem-se à implementação (codificação em uma linguagem de programação), testes e manutenção (mudanças no programa).

### Queremos Desenvolver o Que Mesmo? O Jogo FreeCell

Conforme vimos na Unidade 1, queremos desenvolver como trabalho da disciplina um jogo parecido com o FreeCell (Figura 2.2). Já sabemos *o que* queremos fazer: o FreeCell. Agora estamos na fase de Projeto, e precisamos decidir qual a melhor maneira de desenvolver esse software.

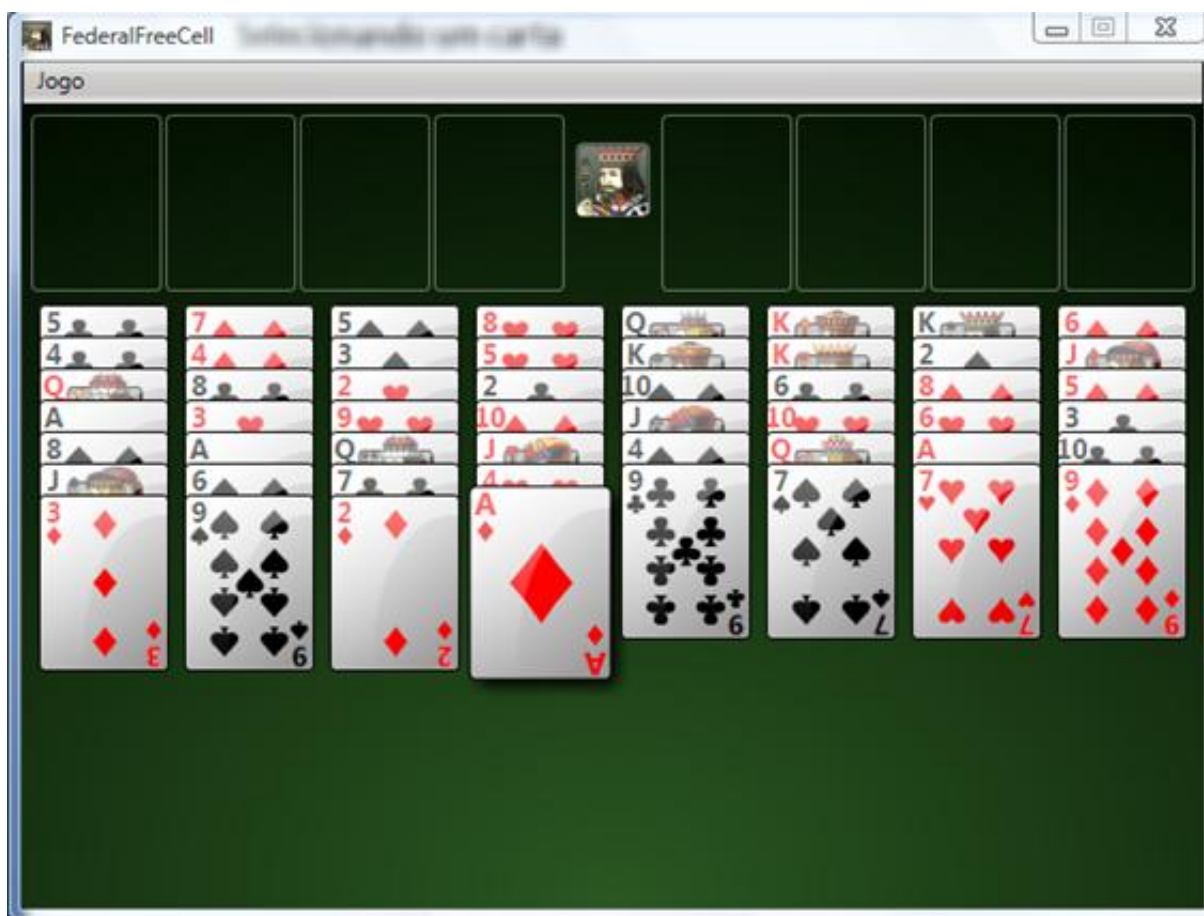


Figura 2.2 Federal FreeCell – desenvolvido por Carlos Eduardo Barbosa, Ronaldo Akio Miyoshi, e Marco Diniz Garcia Gomes, alunos da disciplina Estruturas de Dados, UFSCar, 2008.

Temos, por exemplo, oito pilhas de cartas na parte de baixo da tela. Qual a melhor maneira de lidar com isso na fase de Projeto?

### Definição: Tipo Abstrato de Dados - TAD

Um Tipo Abstrato de Dados – TAD - é uma coleção bem definida de dados a serem armazenados, e um grupo de operadores que podem ser aplicados para manipulação desses dados.



### Características Fundamentais:



- > Os operadores do TAD implementam regras bem definidas para manipulação dos valores armazenados;
- > Os valores armazenados devem ser manipulados **EXCLUSIVAMENTE** pelos operadores do TAD.

A especificação de um TAD descreve **quais dados** podem ser armazenados, e o que é possível fazer com esses dados através dos **operadores** do TAD. Mas a especificação do TAD não descreve **como** isso é ou será efetivamente implementado no programa. Isso pode ser definido em um segundo momento. Ou seja, um Tipo Abstrato de Dado é um modelo abstrato do armazenamento e manipulação de um determinado conjunto de dados de um programa.

### Tipos Abstratos de Dados - Exemplos

Considere os exemplos da Tabela 2.1. No primeiro exemplo, temos um TAD para armazenar e manipular a idade das pessoas. Os dois componentes do TAD são (1) os dados que serão armazenados, e (2) os operadores que podem ser aplicados sobre esses dados: operador *Nasce* e operador *Aniversário*. Uma característica fundamental dos TADs é que os dados armazenados devem ser manipulados **EXCLUSIVAMENTE** através dos operadores do TAD. Nesse caso, isso significa que em nenhum momento será possível, por exemplo, diminuir a idade de uma pessoa no nosso sistema, pois não temos um operador definido para isso.

Mundo Real	Coleção Bem Definida de Dados	Grupo de Operadores
 pessoa	<ul style="list-style-type: none"><li>• a idade da pessoa</li></ul>	<ul style="list-style-type: none"><li>• nasce (idade recebe o valor zero)</li><li>• aniversário (idade aumenta em 1)</li></ul>
 fila de espera	<ul style="list-style-type: none"><li>• nome de cada pessoa e sua posição na fila</li></ul>	<ul style="list-style-type: none"><li>• sai da fila (o primeiro)</li><li>• entra na fila (no fim)</li></ul>
	<ul style="list-style-type: none"><li>• o nome, cargo e o salário de cada funcionário</li></ul>	<ul style="list-style-type: none"><li>• entra no cadastro</li><li>• sai do cadastro</li><li>• altera o cargo</li><li>• altera o salário</li></ul>

 <p>cadastro de funcionários</p>		
 <p>Pilha de Cartas</p>	<ul style="list-style-type: none"> <li>· informações que identificam a carta (nipe, valor), e sua posição na pilha de cartas</li> </ul>	<ul style="list-style-type: none"> <li>· põe uma carta na pilha (no topo da pilha)</li> <li>· retira uma carta da pilha (a do topo)</li> </ul>

**Tabela 2.1 Exemplos de Tipos Abstratos de Dados**

No exemplo da Fila, os operadores permitem que seja retirado apenas o primeiro elemento da fila. E para entrar na Fila, somente no final. Toda fila deveria funcionar realmente assim, certo?

No terceiro exemplo temos um cadastro de funcionários. Que dados serão armazenados? O nome, o cargo e o salário de cada funcionário. Quais os operadores? Operador *Entra no Cadastro*, para o caso de um funcionário ser contratado, *Sai do Cadastro* para o caso de ser despedido, e assim por diante.

O quarto exemplo é uma pilha de cartas. Que informações queremos armazenar? Um conjunto de cartas. Quais as regras básicas do funcionamento de uma pilha de cartas? Regra número 1: só é possível retirar a carta que está no topo da pilha. Regra número 2: só é possível inserir uma carta no topo da pilha. Assim como em uma pilha de pratos: não dá para retirar um prato que não seja o do topo, senão a pilha vai desmoronar. Também não dá para colocar um prato na pilha, a não ser no topo.

Esse tipo abstrato de dados Pilha pode ser útil no projeto do FreeCell?

### O Papel de um TAD na Fase de Projeto do Software

Na fase de Projeto um TAD tem o papel de simplificar. O TAD Cadastro de Funcionários, por exemplo, como vai ser implementado? Em um primeiro momento, na fase de Projeto, a melhor resposta é: não sei! Inicialmente, na fase de Projeto, o que importa sobre o Cadastro é: quais dados serão armazenados, e quais operadores poderão ser aplicados a esses dados. Com isso definido, podemos fazer uma divisão do trabalho entre diversos programadores, por exemplo. Um programador se responsabiliza por implementar o Cadastro. Um outro programador implementa a Folha de Pagamentos, outro implementa a Interface, e assim por diante. E a única coisa que o programador da Folha de Pagamentos conhece do Cadastro é a especificação do TAD: quais dados, quais operadores.

### Exemplo 2.1 – Sistema de Acesso ao Cadastro de Funcionários

Queremos desenvolver um sistema simples para manipulação de um cadastro de funcionários. Temos 3 programadores. O programador 1 irá desenvolver o módulo de Interface. O programador 2 irá desenvolver o TAD Cadastro. O programador 3 ficou de desenvolver um código que utiliza o Módulo Interface e o TAD Cadastro, fazendo o sistema funcionar. Mas o programador 3 não tem a menor idéia de como o cadastro será efetivamente implementado. Só conhece a especificação do TAD Cadastro, e do módulo Interface:

#### Cadastro

Operações e Parâmetros	Funcionamento
------------------------	---------------

Cadastro.InsereNoCadastro( Nome, Cargo, Salário)	Insere no Cadastro um novo conjunto (Nome, Cargo, Salário), passado como parâmetro
Cadastro.RetiraDoCadastro( Nome)	Retira do Cadastro as informações referentes ao Nome passado como parâmetro
Cadastro.NomePertenceAoCadastro?(Nome)	Verifica se o funcionário identificado por Nome, passado como parâmetro, pertence ao Cadastro
Cadastro.ConsultaOSalário( Nome, Salário)	Verifica o salário do funcionário identificado pelo parâmetro Nome, retornando o valor da salário através do parâmetro Salário.

## Interface

Operações e Parâmetros	Funcionamento
Interface.QueOperaçãoQuerFazer?(Operação, Nome, Salário)	O usuário escolhe uma Operação a ser executada no Cadastro. As escolhas do usuário retornam no parâmetro Operação e, quando pertinente, nos parâmetros Nome e Salário
Interface.AíVaiAResposta(TextoResposta)	Mostra ao usuário o Texto Resposta passado como parâmetro.

Sem ter a menor idéia de como o Cadastro e a Interface são efetivamente implementados, o programador 3 desenvolveu o seguinte código:

Repita-para-Sempre:

Interface.QueOperaçãoQuerFazer?(Operação, Nome, Salário)

Caso Operação for..

```

'inserir': Se Cadastro.NomePertenceAoCadastro?(Nome) = verdade
    então Interface.AíVaiAResposta('Esse nome já está no cadastro! Não dá para inserir de novo.')
    senão Cadastro.InsereNoCadastro( Nome, Cargo, Salário)
           Interface.AíVaiAResposta('Funcionário inserido com sucesso!')

'retirar': Se Cadastro.NomePertenceAoCadastro?(Nome) = verdade
    então Cadastro.RetiraDoCadastro( Nome)
           Interface.AíVaiAResposta('Funcionário retirado do cadastro com sucesso!')
    senão Interface.AíVaiAResposta('Esse funcionário já não estava no cadastro!')

'consulta salário': Se Cadastro.NomePertenceAoCadastro?(Nome) = verdade
    então Cadastro.ConsultaOSalário( Nome, Salário)
           Interface.AíVaiAResposta('Salário = ', Salario)
    senão Interface.AíVaiAResposta('Não achei esse funcionário!')
```

fim do Caso

fim do Repita-para-Sempre

Note, nesse algoritmo, que a manipulação dos dados armazenados no Cadastro é feita exclusivamente pelas operações definidas na especificação do TAD. O programador 3 não tem a menor ideia de como essas operações são implementadas. Mas, manipulando o Cadastro exclusivamente através das operações, realmente não é preciso saber detalhes de implementação.

## Curiosidade Sobre Tipos Abstratos de Dados: O Que Significa o termo *Abstrato*?

--

**Abstrair:**

[1] considerar isoladamente um ou mais elementos de um todo; separar, apartar

[4 - filós] separar mentalmente para tomar em consideração (uma propriedade que não pode ter existência fora do todo concreto ou intuitivo em que aparece): abstrair a cor ou a forma de um objeto

**Abstrato:**

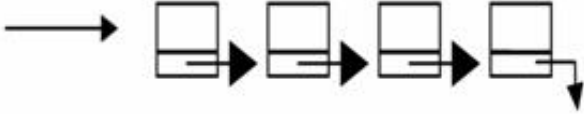

[5 - filós] diz-se de representação à qual não corresponde nenhum dado sensorial ou concreto...

**Abstração:**

[2 - filós] ato de separar mentalmente um ou mais elementos de uma totalidade complexa (coisa, representação, fato), os quais só mentalmente podem subsistir fora dessa realidade.

fonte: Dicionário Aurélio

No contexto de nossa disciplina, Estruturas de Dados, vamos abstrair (considerar isoladamente, representar) elementos do mundo real. Vamos trabalhar com desenhos (representações) de estruturas (veja uma representação que utilizaremos na disciplina na Tabela 2.2). Essas representações só existem em nossa imaginação. Ou seja, as representações são abstratas, não correspondentes a nenhum dado sensorial ou concreto. Na biologia podemos trabalhar com elementos concretos, seres vivos, os quais podemos pegar e cheirar. Os desenhos que faremos nessa disciplina são representações abstratas, ou seja, jamais conseguiremos cheirar ou pegar o que eles representam. Os zeros e uns, elementos básicos da computação, são abstratos; só existem em nossa imaginação.

<b>FUNCIONALIDADE</b> Cadastro.InsereNoCadastro( Nome, Cargo, Salário) Cadastro.RetiraDoCadastro( Nome) Cadastro.NomePertenceAoCadastro?(Nome) Cadastro.ConsultaOSalário( Nome, Salário)	<b>IMPLEMENTAÇÃO</b> <pre>void push( int x, int &amp;erro) {     if (topo==StackSize-1)         erro = 1;     else {         erro = 0;         topo++;         Itens[topo] = x; } }</pre>
 <b>REPRESENTAÇÃO ABSTRATA: SÓ EXISTE EM NOSSA IMAGINAÇÃO</b>	 <b>MUNDO REAL</b>

**Tabela 2.2 Mundo real, funcionalidade, representação abstrata e implementação.**

Nessa disciplina procuraremos também abstrair, no sentido de fazer uma separação mental entre funcionalidade, representação e implementação. Em determinados momentos procuraremos trabalhar apenas com a funcionalidade de um Tipo Abstrato de Dados, fazendo o possível para esquecer como se dá (ou se dará) sua implementação. Em determinados momentos procuraremos nos concentrar exclusivamente na representação, no desenho, para não errarmos na elaboração do código.

**Qual a Melhor Maneira de Aumentar o Volume da TV?**

Podemos aumentar o volume da TV acionando o botão do volume, ou podemos abrir a TV com uma chave de fenda, buscando alterar na marra um componente eletrônico para aumentar o volume. Qual alternativa você acha mais interessante?

Em uma analogia, podemos considerar um Tipo Abstrato de Dados como uma TV. Podemos desenvolver programas aumentando o volume através do botão da TV, ou podemos desenvolver programas aumentando o volume com uma chave de fenda. Aumentar o volume pelo botão da TV significa utilizar Tipos Abstratos de Dados, definir dados a serem armazenados, e operações aplicáveis a esses dados, e a partir daí só manipular os dados através desses operadores. Os operadores, no caso, equivalem aos botões da TV (Tabela 2.3).



Mundo Real	Coleção Bem Definida de Dados	Grupo de Operadores
 Pilha de Cartas	<ul style="list-style-type: none"><li>· informações que identificam a carta (nipe, valor), e sua posição na pilha de cartas</li></ul>	<ul style="list-style-type: none"><li>· põe uma carta na pilha (no topo da pilha)</li><li>· retira uma carta da pilha (a do topo)</li></ul>
 TV (apenas para analogia)	<ul style="list-style-type: none"><li>· som, imagem</li></ul>	<ul style="list-style-type: none"><li>· aumentar o volume</li><li>· diminuir o volume</li><li>· mudar de canal (canal acima)</li><li>· mudar de canal (canal abaixo)</li></ul>

Tabela 2.3 Tipos Abstratos de Dados – Analogia com TV

Alterar os dados de um cadastro sem ser por uma operação definida na especificação do TAD é possível? É possível, mas fazendo isso estamos fazendo o equivalente a abrir uma TV com uma chave de fenda para aumentar o volume. Manipular uma pilha de cartas sem ser por operações precisamente definidas na especificação do TAD Pilha é possível? É possível também. Estaremos aumentando o volume com uma chave de fenda, abrindo a TV, nesse caso também.

O Que É um Bom Programa?

Não, não é um programa que funciona. Se não funcionar, não é um programa. Vamos mudar a pergunta: temos 2 programas. Os 2 funcionam. Um deles é um bom programa, e o outro é um programa ruim. Quais as características do programa que é bom?

Temos vários critérios para diferenciar dois programas: qual roda mais rápido, qual usa menos memória e assim por diante. Temos alguns critérios de qualidade muito importantes, que são:

- **Reusabilidade de código:** a capacidade de reutilizar um determinado código (programa), em uma segunda situação;



- **Portabilidade de código** (o que é um computador portátil? E um código portátil?): a capacidade de um código (programa) executar em diferentes ambientes (plataformas) de hardware e software. Ou seja, assim como podemos carregar um computador portátil de um lugar para o outro, podemos carregar um software portátil de uma plataforma para outra, e ele continuará funcionando.

Se tivermos que escolher entre um código com maior portabilidade / reusabilidade, e um código que execute milésimos de segundo mais rápido, o que você escolheria? Pense um pouco sobre isso...

Na maior parte das circunstâncias, é muito melhor optar pela portabilidade/reusabilidade, pois isso resultará em um custo muito mais baixo do software, em especial custo de manutenção. Você concorda com isso?

#### **Atividade Complementar: Pesquise e Opine**

Faça uma pesquisa na internet sobre os termos “portabilidade software” e também “reusabilidade software”. Leia alguns artigos sobre o assunto. Conte aos colegas o que achar de interessante sobre isso. Você concorda que portabilidade e reusabilidade são critérios, em geral, mais importantes do que milésimos de segundo? Emita sua opinião!

### **Vantagens da Utilização do Conceito de Tipos Abstratos de Dados**

- Mais fácil programar, sem se preocupar com detalhes de implantação. No momento de transferir dados de uma pilha para outra, você não precisa nem se incomodar em saber como a pilha é efetivamente implementada. Você precisa apenas saber utilizar as operações (ou botões) da pilha.
- Mais seguro programar: apenas as operações do Tipo Abstrato de Dados alteram os dados. As operações do TAD Cadastro, desenvolvidas pelo programador 2, estão corretas, funcionam bem, pois o programador 2 conhece bem a implementação do Cadastro. Se o programador 3, que não conhece bem a implementação do Cadastro, for alterar os dados armazenados *com a chave de fenda* (ou seja, sem usar os botões da TV - as operações do TAD), é possível que ele faça alguma besteira com os dados.
- Maior independência e portabilidade de código: alterações na implementação de um TAD não implicam em alterações em seu uso. Vamos supor que implementamos um TAD Cadastro de um determinado modo, mas depois decidimos, por algum motivo, mudar a implementação. Se uma aplicação que usa o cadastro tiver sido implementada apenas através do acionamento dos operadores do cadastro, como ocorreu no Exemplo 2.1, essa aplicação continuará funcionando, ainda que a implementação do TAD Cadastro mude. O que mudou foi a implementação do Cadastro, mas sua funcionalidade (seus operadores) continuam os mesmos!
- Maior potencial de reutilização de código: pode-se alterar a lógica de um programa sem a necessidade de reconstruir as estruturas de armazenamento. Um mesmo TAD cadastro pode ser utilizado em um sistema de folha de pagamentos, e também em um sistema de controle de frequência.
- A consequência de todas essas vantagens: menor custo no desenvolvimento de software.

### **Exercícios de fixação**

1. Que é um [Tipo Abstrato de Dados – TAD](#)? Quais são as características fundamentais?
2. O que é [portabilidade de código](#)? O que é [reusabilidade de código](#)?
3. Quais as [vantagens](#) de se programar utilizando o conceito de TADs? Explique com exemplos.
4. [Como é possível desenvolver um programa utilizando um TAD Cadastro, por exemplo, sem conhecer detalhes de sua implementação?](#)
5. Caso ainda não tenha feito, faça uma [pesquisa sobre portabilidade e reusabilidade](#), e converse com os colegas sobre o que achou importante sobre isso. Você concorda que portabilidade e reusabilidade são, na maioria das situações, mais importantes do que um programa executar milésimos de segundo mais rápido?

### **Pensamento do Dia**



*Nunca desmonte uma TV para aumentar o volume; use o botão.*

