

Introdução

Prof. Marcos Carrard
carrard@gmail.com

Conteúdo

- Objetivos
- Tipo abstrato de dados
- Estrutura de dados
- Vantagens e características dos TADS
- Estudo de caso



Objetivos

- Entender o conceito de Tipos Abstratos de Dados, e o modo de utilizá-lo no desenvolvimento de programas;
- Perceber que o uso de Tipos Abstratos de Dados dá ao software maior portabilidade, maior potencial para reutilização, reduz custos de desenvolvimento e de manutenção;
- Conscientizar-se quanto a importância de adotar uma estratégia que agregue portabilidade e reusabilidade aos jogos que você desenvolverá.
- Diferenciar Tipos Abstratos de Dados e Estrutura de Dados



Tipos de dados e tipos abstratos de dados

Tipo de Dados

- Em linguagens de programação o tipo de dado de uma variável, constante ou função define o conjunto de valores que a variável, constante ou função podem assumir.
- Exemplo: int, float, etc..

TADS

- Um Tipo Abstrato de Dados – TAD - é constituído por um conjunto de **Dados** a serem armazenados, e por um grupo de **Operadores** que podem ser aplicados para manipulação desses Dados.
- O armazenamento e a recuperação dos Dados devem ser realizados **exclusivamente** através dos Operadores do TAD.



Estruturas de dados

Dado

- Tipos de dados:
- Inteiro (int)
 - Texto (string)
 - Caracter (char)
 - Ponto flutuante (float)
 - Ponto flutuante (double)

Estrutura

- Estruturas:
- Vetores multidimensionais
 - Pilhas
 - Filas
 - Listas

Estrutura de dados é o ramo da computação que estuda os diversos mecanismos de organização de dados para atender diferentes requisitos de processamento.



Vantagens dos TADS

- Separação entre o conceito (definição do tipo) e a implementação das suas operações;
- O acesso e visibilidade a estrutura de representação do dados é restrita as operações que o manipulam;
- As aplicações (programas) que farão uso do TAD atuam como clientes de um serviço, sem preocupar-se como ele foi implementado, mas tendo a garantia de que isso aconteceu de forma correta.
- Por consequência, esse tipo de implementação facilita o reuso do código, sua manutenção e correção, todos aspectos de implementação e que não devem afetar os seus clientes.




TADs – Tipo Abstrato de Dados

- Abstração: é a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes.
- Ao definirmos um TAD, nos concentramos nos aspectos essenciais do tipo de dado (operações) e nos abstraímos de como ele foi implementado.



Exemplo: *FreeCell*

TAD Pilha Intermediária do <i>FreeCell</i>		
	Coleção de Dados a Serem Armazenados	Operadores Para Manipulação
	<p>Para cada uma das <i>Pilhas Intermediárias</i>:</p> <ul style="list-style-type: none"> As cartas que estão na pilha - valor e naipe de cada carta; A sequência das cartas na pilha. 	<ul style="list-style-type: none"> Retira a carta que está no topo da pilha; Coloca uma carta no topo da pilha, se o valor estiver na sequência correta.



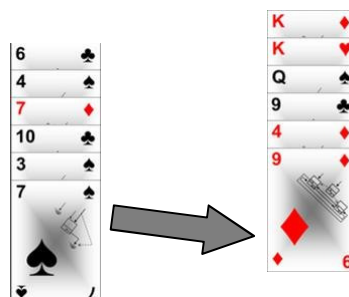
Operações do TAD Pilha Intermediária



- Desempilha (Pilha, Carta, DeuCerto)
- EmpilhaNaSequência (Pilha, Carta, DeuCerto)
- EmpilhaSempre (Pilha, Carta)



Exercício 1.1 Transfere Carta



TransfereCarta (parâmetros por referência **PilhaOrigem**, **PilhaDestino** do tipo **PilhaIntermediária**, parâmetro por referência **DeuCerto** do tipo **Boolean**)

/* Transfere uma carta da Pilha Origem para a PilhaDestino, caso a carta estiver na sequência correta na PilhaDestino. O parâmetro DeuCerto retornará o valor Verdadeiro se uma carta for efetivamente transferida, e o valor Falso caso contrário */



TransfereCarta (parâmetros por referência **PilhaOrigem**, **PilhaDestino** do tipo **PilhaIntermediária**, parâmetro por referência **DeuCerto** do tipo **Boolean**) {

Variável Carta do tipo Carta-do-Baralho;

Variável ConseguiuRetirar do tipo Boolean;

Variável ConseguiuEmpilhar do tipo Boolean;

/* Tenta retirar Carta do topo da PilhaOrigem */

Desempilha(PilhaOrigem, Carta, ConseguiuRetirar);

Se (ConseguiuRetirar = Verdadeiro)

Então { /* empilha na Pilha Destino, se estiver na sequência correta */

EmpilhaNaSequência(PilhaDestino, Carta, ConseguiuEmpilhar);

Se (ConseguiuEmpilhar = Verdadeiro)

Então DeuCerto = Verdadeiro;

Senão { /* carta não está na sequência correta e deve retornar à PilhaOrigem */

EmpilhaSempre(PilhaOrigem, Carta);

DeuCerto = Falso;

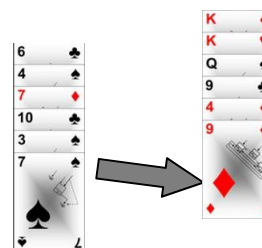
}

}

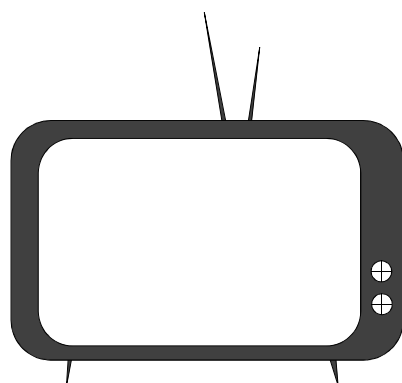
Senão

DeuCerto = Falso;


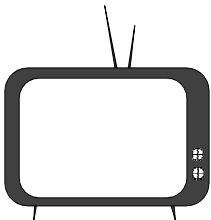
}



Qual a Melhor Maneira de Aumentar o Volume da TV?



Qual a Melhor Maneira de Aumentar o Volume da TV?

Operadores do TAD ou Botões da TV	
	Pilha de Cartas: <ul style="list-style-type: none"> Retira a carta que está no topo da pilha; Coloca uma carta no topo da pilha, se o valor estiver na sequência correta.
	TV: <ol style="list-style-type: none"> Aumenta o volume; Diminui o volume; Muda de canal (1 canal acima); Muda de canal (1 canal abaixo).



O Que É um Bom Programa?

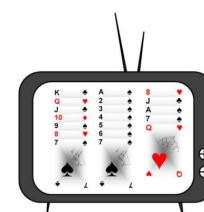
Portabilidade de Software: capacidade de executar em diferentes plataformas de hardware e software.

Reusabilidade de Software: capacidade de aproveitar (reutilizar) um software já desenvolvido, para satisfazer uma segunda necessidade.



Vantagens da Utilização de Tipos Abstratos de Dados

- É mais fácil programar, sem se preocupar com detalhes de implantação;
- É mais fácil preservar a integridade dos dados;
- Maior independência e portabilidade de código;
- Maior potencial de reutilização de código.



Software Bom, Bonito e Barato

O uso do conceito de Tipos Abstratos de Dados aumenta a Portabilidade e o potencial de Reutilização do software. Em consequência disso, o custo de desenvolvimento e manutenção é reduzido.



Como definir um TAD

- programador descreve o TAD em dois módulos separados
- Um módulo contém a definição do TAD: representação da estrutura de dados e implementação de cada operação suportada
- Outro módulo contém a interface de acesso: apresenta as operações possíveis
- Outros programadores podem, por meio da interface de acesso, usar o TAD sem conhecer os detalhes representacionais e sem acessar o módulo de definição



Implementação de um TAD

- Uma vez definido um TAD e especificadas as operações associadas, ele pode ser implementado em uma linguagem de programação
- Uma estrutura de dados pode ser vista, então, como uma implementação de um TAD
 - implementação do TAD implica na escolha de uma ED para representá-lo, a qual é acessada pelas operações que ele define
- ED é construída a partir dos tipos básicos (integer, real, char) ou dos tipos estruturados (array, record) de uma linguagem de programação



Estudo de Caso





K-HARD

