1. Introdução a Estrutura de Dados

Tipos de Dados e Estruturas de Dados

- Aplicação para Engenharia de Computação é um programa de computador que manipula dados
- A representação destes dados é feita por estruturas de dados
- Quão bem as estruturas de dados coincidem com um problema? (Temos os tipos de dados da linguagem… são suficientes?)

Tipos de Dados e Estruturas de Dados

- Tipos de dados primitivos não pode ser decomposto estruturalmente (inteiros, decimais, caracteres, lógicos, …)

- Quão bem as estruturas de dados coincidem com um problema?
  As linguagens, costumeiramente, possuem tipos de dados adequáveis (*structs*, *records*, *registros, classes, …*)

Tipos de Dados e Estruturas de Dados

- Tipos de Dados Estruturados
  - agregam mais de um tipo primitivo
  - arranjos, seqüências, …
- Tipos de Dados Definidos pelo Usuário (são Tipos de Dados Estruturados)
  - construídos hierarquicamente
  - tipo construído pelo usuário é constituído por diversos componentes, que podem ser de tipos diferentes, sob um único tipo

Tipos Abstratos de Dados (TAD)

- São estruturas de dados capazes de representar tipos de dados não previstos na criação das linguagens de programação
- Necessárias para aplicações específicas

Tipos Abstratos de Dados (TAD)

- Diferenciação: Conceito e Aplicação, ou seja, diferenciação na definição do tipo e sua representação, e a implementação das operações na linguagem
- Exemplo: tipo Data
  - quantos valores armazenamos?
  - operações sobre a data (qtos dias de diferença entre uma data e outra)

Tipos Abstratos de Dados (TAD)

- Exemplo: tipo Data
  - formalmente podemos representar a TAD de Data pelo par (v, o)
    - v: tripla de informações dia-mês-ano
    - o: operações realizadas sobre a data (inicializar data, acrescentar dias e escrever data por extenso)
- Uma vez que o TAD foi caracterizado, próximo passo é escolher a estrutura de representação

Tipos Abstratos de Dados (TAD)

- Código:

```
struct Data{
    int dia;
    int mes;
    int ano;
};
```

Notem que a representação é obtida através de uma coleção de tipos primitivos, ou mesmo uma estrutura complexa formada por diversos tipos primitivos.

```
struct {
    int dia;
    int mes;
    int ano;
};
```

Tipos Abstratos de Dados (TAD)

- Além da estrutura de dados, uma TAD envolve operações (funções e procedimentos) que podem ser executados com este novo tipo de dados!

Tipos Abstratos de Dados (TAD)

- Exemplo: tipo Data
  - InicializaData
    - Entradas: dia, mês, ano (inteiro)
    - Saída: d (Data)
  - AcrescentaDias
    - Entradas: data (Data), dias (inteiro)
    - Saída: Data
  - EscrevaExtenso
    - Entradas: data (Data)
    - Retorno: v (Vazio)

Tipos Abstratos de Dados (TAD)

- Deste modo, um TAD é composto por uma definição abstrata da estrutura de dados e pela especificação das operações aplicáveis sobre ela
- A representação em linguagem de programação é a *instanciação* de uma TAD
- Esta separação da implementação permite que a mesma TAD possa ser aplicada a outras linguagens

Tipos Abstratos de Dados (TAD)

- Quando se implementa as operações de um TAD é importante utilizar-se de conceitos de bibliotecas
- Deste modo, a implementação da TAD fica em uma biblioteca, separado da módulo principal da aplicação

CLASSIFICAÇÕES

Tipos Abstratos de Dados (TAD)

- Termos e Classificações …:
  - Especificação da TAD: tipo abstrato
  - Implementação da TAD: tipo concreto
  - TAD específica: tipo de dado fixo
  - TAD genérica: tipo de dados configurável (por exemplo, listas de alunos, de números, de datas)

Representação Física

- Contigüidade
  - os dados são armazenados em posições contíguas na memória
  - cada posição contígua na memória armazena informações correspondente a um nodo

Representação Física

- Contigüidade
  - Vantagens:
    - proteção de memória: alocação é feita antes do início da execução
    - transferência de dados: como todos os dados estão alocados em bloco, transferência da memória principal e secundária é facilitada
    - estruturas simples: apropriada para armazenar estruturas simples

Representação Física

- Contigüidade
  - Vantagens:
    - representação: em estrutura simples é vantajosa
    - acesso: qualquer nodo pode ser associado a qualquer momento, através de um índice associado a posição

Representação Física

- Contigüidade
  - Desvantagens:
    - compartilhamento de memória: não permite
    - previsão de espaço físico: deve-se se definir, antes da execução, o limite máximo de nodos
    - estruturas complexas: não apropriado para estruturas complexas
    - inserção e exclusão de componentes: geralmente implicam deslocamento de um número considerável de componentes

Representação Física

- Encadeamento
  - alocação dinâmica
  - permite forma de alocação do tipo ponteiro

Representação Física

- Encadeamento
  - Vantagens
    - compartilhamento de memória: os nodos podem fazer parte de mais de uma estrutura
    - maleabilidade: alocação e liberação de memória favorece a maleabilidade dos programas
    - facilidade para inserção e remoção de componente

Representação Física

- Encadeamento
  - Desvantagens
    - tranferência de dados: dificultada
    - gerência de memória mais onerosa: alocação dinâmica é realizada através do gerenciador de memória
    - procedimentos menos intuitivos: procedimentos mais complexos
    - acesso: um nodo pode somente ser acessado através de outros

Exercícios

Exercícios

- Considere uma aplicação para armazenar dados de um jogo de cartas.
  - Teríamos que tipos de TADs?
  - Quais operações seriam utilizadas?