Estruturas de Dados

João Paulo Dias de Almeida
joao.almeida@academico.ifs.edu.br

Instituto Federal de Sergipe

O que vamos aprender hoje?

- Entender a importância do estudo de Estruturas de Dados
- Descrever Tipos Abstratos de Dados
- Conhecer alguns dos principais modelos de dados
- Diferenciar dado contíguo de dado ligado

Introdução

- Coleções de dados estão presentes na grande maioria dos programas de computador
  - Por isto são tão importantes para o estudo da Computação
- Algoritmos podem manipular a coleção, fazendo-a crescer, encolher, ou seja alterada
  - Conjunto dinâmico

Introdução

- Cada algoritmo vai exigir operações diferentes a serem executadas em uma coleção
  - Por exemplo, em um dicionário, exige-se apenas a habilidade de inserir elementos, buscar, e deletar
- Outros algoritmos podem exigir operações mais complexas
  - Extrair o menor elemento de um conjunto
- A melhor maneira de implementar uma coleção de dados depende das operações que precisam ser suportadas

“Trocar uma estrutura de dado em um programa lento pode funcionar da mesma maneira que um transplante em um paciente doente” – Steven Skiena

“Changing a data structure in a slow program can work the same way an organ transplant does in a sick patient.” – Steven Skiena

Definições

Uma estrutura de dados pode ser dividida em dois pilares fundamentais: **dado** e **estrutura**

Dado	Estrutura
Elemento que possui valor agregado e que pode ser utilizado para solucionar problemas computacionais. Os dados possuem tipos específicos	Elemento estrutural que é responsável por carregar as informações em um software

Definições

Dado

Tipos de dados:

- Inteiros (int)
- Reais (float e double)
- Texto (String)
- Caracter (char)

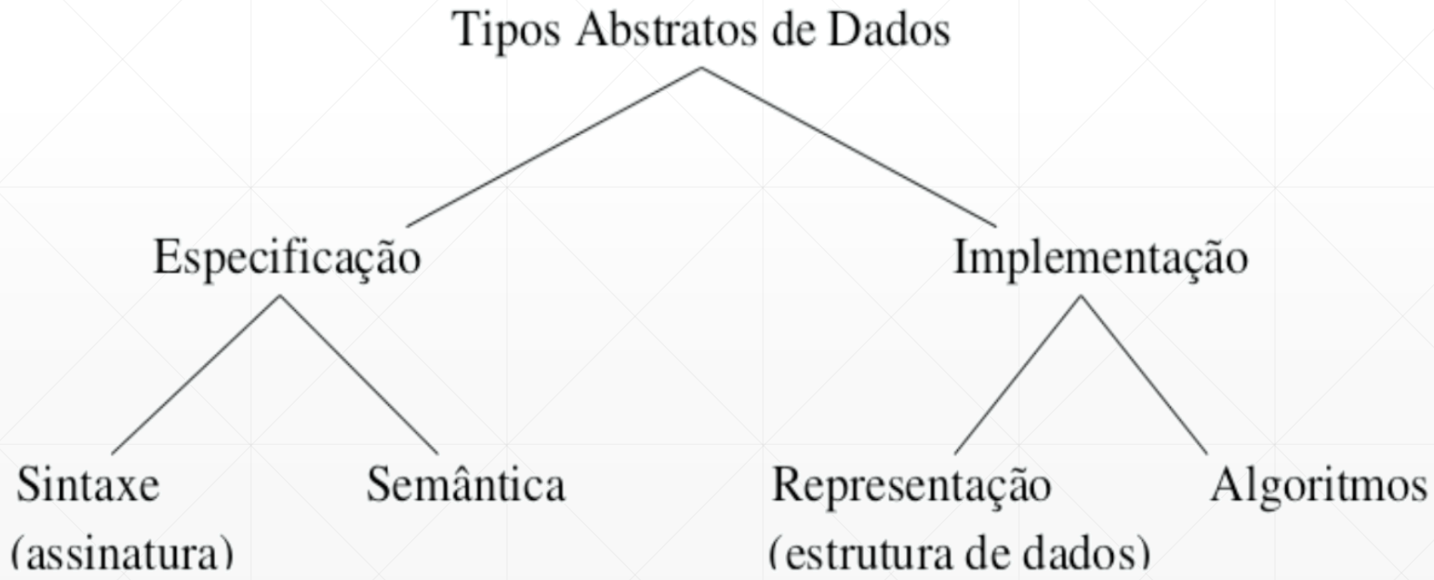
Estrutura

Estruturas:

- Arrays multidimensionais
- Pilhas
- Filas
- Listas
- Árvores

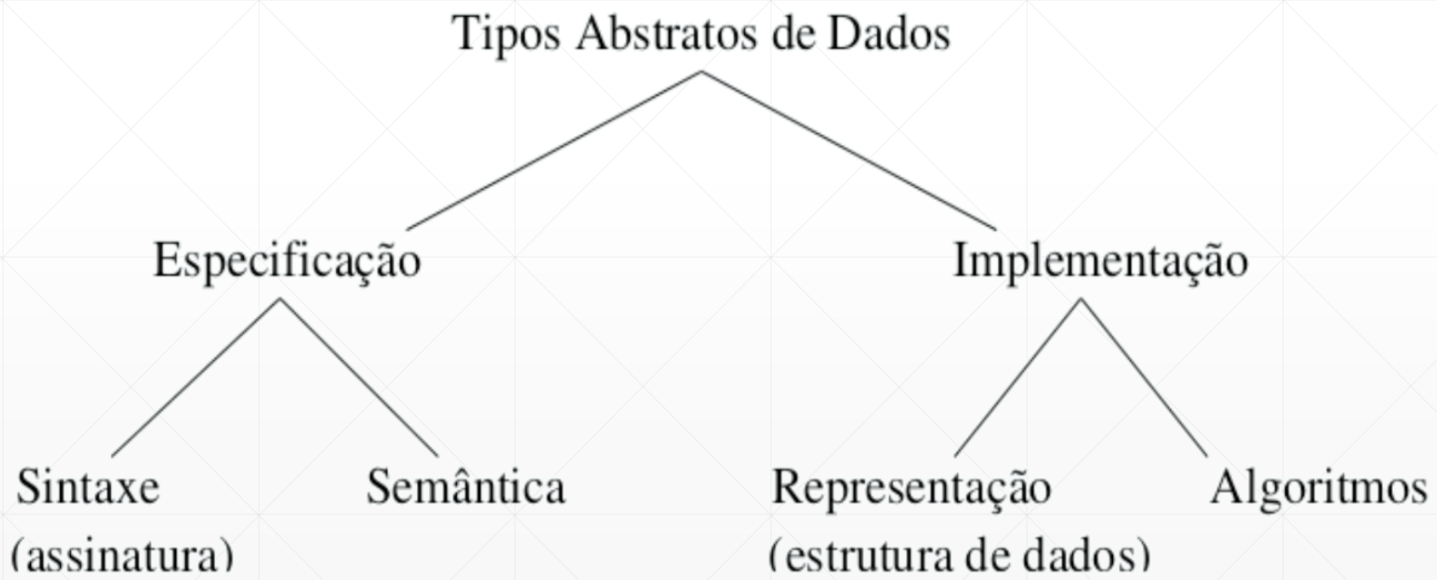
Tipo abstrato de dado (TAD)

- Separa a especificação da implementação
  - É possível usar a estrutura sem saber detalhes de como ela funciona



Tipo abstrato de dado (TAD)

- O mesmo TAD pode possuir diferentes implementações



Tipo abstrato de dado (TAD)

- A sintaxe de um TAD é um conjunto de assinaturas que especifica a sua interface

- A especificação sintática de um TAD pode não ser suficiente para descrever o comportamento
  - Especificar a sua semântica de maneira independente da implementação

Tipo abstrato de dado (TAD)

- Abstração é a habilidade de focar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes

- Quando definimos um TAD, nos concentramos nos aspectos essenciais do tipo de dado (operações) e nos abstraímos de como ele foi implementado

Tipo abstrato de dado (TAD)

O Tipo Abstrato de Dado (TAD) é uma especificação de um conjunto de dados e operações que podem ser executadas sobre esses dados

Exemplo: Tipo abstrato de dado (TAD)

Objetivo: inserir um número no início da lista

Implementação por Vetor:


```
1 void Insere(int x, Lista L) {
2   for(i=0;...) {...}
3   L[0] = x;
4 }
```

Implementação por Lista Encadeada:


```
1 void Insere(int x, Lista L) {
2   p = CriaNovaCelula(x);
3   L.primeiro= p;
4   ...
5 }
```

Programa do usuário da TAD:

```
1 int main() {
2   Lista L;
3   int x;
4   x = 20;
5   FazListaVazia(L);
6   Insere(x,L);
7   ...
8 }
```

Implementação: TAD

- Em linguagens orientadas a objeto, a implementação é feita usando classes e a especificação usando interfaces
  - Em Java, em possível reutilizar implementações já prontas
- Em linguagens estruturadas (C, pascal) a implementação é feita pela definição de tipos juntamente com a implementação de funções.
  - typedef e structs

Exemplo – TAD reutilizado

Exemplo: TAD personalizado

Elementos de um conjunto dinâmico

- Cada elemento costuma ser representado por um objeto
  - Cada atributo do objeto pode ser acessado e manipulado
- É comum que um dos atributos seja uma chave identificadora do elemento
- O objeto pode conter dados satélite que são associados a outros atributos mas que não são utilizados pela implementação da estrutura

Exemplo: chave e dados satélite

key	c1	c2	c3	c4	c5
2	“nome2”	“cpf2”	“end2”	“tel2”	“sal2”
3	“nome3”	“cpf3”	“end3”	“tel2”	“sal2”
1	“nome1”	“cpf1”	“end1”	“tel1”	“sal1”
3	“nomex”	“cpfx”	“endx”	“telx”	“salx”
4	“nome4”	“cpf4”	“end4”	“tel4”	“sal4”
5	“nome5”	“cpf5”	“end5”	“tel5”	“sal5”

Operações comuns

- Operações podem ser agrupadas em duas categorias: buscas (*queries*) e operações modificadoras
- Lista de operações típicas:
  - **Buscar(S, k)**

    Dado um conjunto S e um valor k, retorna uma referência para o elemento em S que possui o valor k
  - **Inserir(S,x)**

    Insere o elemento x no conjunto S. x deve possuir os mesmos atributos que os demais elementos de S

Operações comuns

- Operações podem ser agrupadas em duas categorias: buscas (*queries*) e operações modificadoras
- Lista de operações típicas:
  - **Deletar(S,x)**

    Remove x do conjunto S
  - **Mínimo(S)**

    Retorna o elemento de S que possui a menor chave. Pressupõe que S está ordenado

Operações comuns

- Operações podem ser agrupadas em duas categorias: buscas (*queries*) e operações modificadoras
- Lista de operações típicas:
  - **Máximo(S)**

    Retorna o elemento de S que possui a maior chave. Pressupõe que S está ordenado

Estruturas de Dados

Estruturas de Dados

- Arrays
- Dicionários
- Pilha e Fila
- Lista encadeada
- Tabela Hash
- Árvore

Dados contíguos vs Dados ligados

- Estruturas de dados podem ser classificadas como contíguas ou ligadas
  - Contíguas → baseadas em arrays
  - Ligadas → baseadas em ponteiros/referências
- Estruturas alocadas de forma contígua são compostas por blocos de memórias únicos.
  - Arrays e matrizes
  - Pilha
  - Tabela Hash

Dados contíguos vs Dados ligados

- Estruturas de dados ligados são compostas por blocos de memórias distintos que são unidos pelo uso de ponteiros
  - Listas
  - Árvores
  - Grafos

Referências

- CORMEN, Thomas. Desmistificando Algoritmos. Editora Campus, 2012.
- Fortes, R. TAD: Tipo Abstrato de Dado. DECOM-UFOP. Disponível em: https://bit.ly/3F2edVk . Acessado em: 15/12/2021.
- Rocha-junior, J. B. Força Bruta: ordenação. Notas de aula. Disponível em: https://bit.ly/2YvYWMJ. Acessado em: 09/11/2021
- SKIENA, Steven. The Algorithm Design Manual. 6ª edição. Springer, 2020.

Dúvidas?