Grafos - Tipo Abstrato de Dados (TAD)

Prof. Luiz Gustavo Almeida Martins

Tipo Abstrato de Dados (TAD)

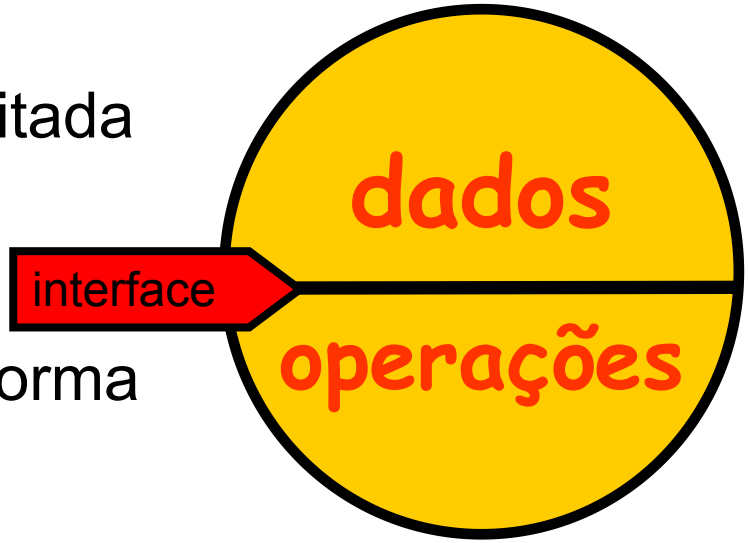
Forma de definir um **novo** **tipo de dado** e as **operações** que o manipulam

Baseada na definição de **tipos estruturados**

Ideia central: encapsular (**esconder**) de quem usa o TAD a forma como ele foi implementado

– Visibilidade da estrutura fica limitada às operações

– Cliente tem acesso somente à forma abstrata do TAD



Tipo Abstrato de Dados (TAD)

Um TAD é definido por:

Um **conjunto de valores** (dados)

- Atributos/campos da estrutura
- Define o que a estrutura deve representar

Um **conjunto de operações** que atuam sobre esses valores

- Determinam as **ações** realizadas na manipulação dos dados do TAD
- Devem ser consistentes com os tipos utilizados

Tipo Abstrato de Dados (TAD)

Permite a separação entre o conceito (**o que fazer**) e a implementação (**como fazer**)

Vantagens:

Encapsulamento e segurança:

Somente o conceito do TAD é visível externamente

Usuário **NÃO** tem acesso direto aos dados

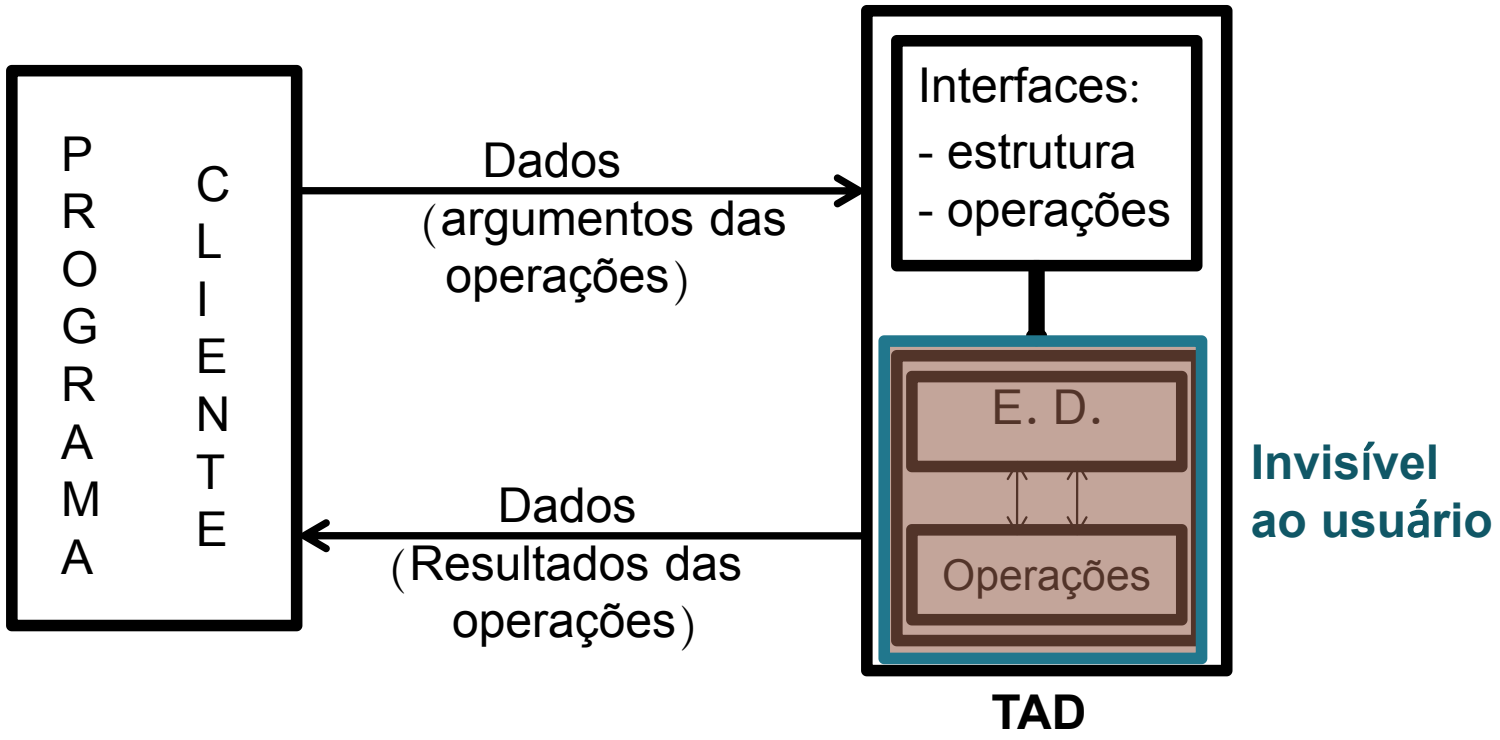
Acesso somente através das operações

Flexibilidade e reutilização:

Compilação separada: mudança na implementação do TAD não afeta o programa que o utiliza (programa usuário ou cliente)

As interfaces das operações devem ser mantidas

Iteração entre TAD e programa usuário



Especificação de um TAD

Define a **parte conceitual** do TAD (**o que deve ser feito**)

Determina as interfaces

Deve conter as seguintes informações:

Cabeçalho:

Nome: identificação do TAD

Dados: descrição dos tipos dos dados da estrutura

Lista das operações: nome das operações que manipulam a estrutura

Especificação das operações:

Entradas: informação necessária para executar a operação

Pré-condição: verificada antes de executar a operação

Processo: tarefas que devem ser realizadas para realizar a operação

Saída: valor resultante do processamento

- Retornado explicitamente pela operação

Pós-condição: indica alterações na estrutura após a operação

- Retorno implícito (alteração em uma variável passada por referência)

Estrutura Geral da Especificação

TAD **nome_TAD**:

Dados: descrição dos campos da estrutura de dados

Lista de operações: operação1, operação2, ..., operação *N*

Operações:

Operação1:

Entrada:
Pré-condição:
Processo:
Saída:
Pós-condição:

Operação2:

Entrada:
Pré-condição:
Processo:
Saída:
Pós-condição:

...

Operação *N* :

Entrada:
Pré-condição:
Processo:
Saída:
Pós-condição:

Especificação do TAD Dígrafo

Dígrafo = **Grafo Direcionado**

TAD **Dígrafo**

Dados: chave identificadora (número inteiro)

Lista de operações: cria_grafo, insere_aresta, verifica_aresta, remove_aresta, consulta_aresta, libera_grafo, mostra_adjacentes, mostra_grafo

Especificação do TAD Dígrafo

Operação ***cria_grafo:***

Entrada: a quantidade de vértices

Pré-condição: quantidade de vértices ser válida

Processo: alocar a área para representar o grafo, se necessário, e colocá-lo na **condição de vazio**

Saída: o endereço do grafo se operação bem sucedida ou ***NULL*** se houve algum erro

Pós-condição: nenhuma

Especificação do TAD Dígrafo

Operação ***insere_aresta:***

Entrada: o endereço do grafo, os identificadores do par de vértices (V_i e V_j), e o peso da aresta (P)

Pré-condição: o grafo existir e os vértices serem válidos e a aresta não existir

Processo: inserir uma aresta do vértice de origem (V_i) para o vértice de destino (V_j) com peso P

Saída: 1 se sucesso, 0 se aresta já existe ou -1 se grafo inconsistente

Pós-condição: o grafo de entrada com uma nova aresta

Especificação do TAD Dígrafo

Operação ***verifica_aresta:***

Entrada: o endereço do grafo e os identificadores do par de vértices (V_i e V_j)

Pré-condição: o grafo existir e os vértices serem válidos

Processo: verifica se existe aresta entre o vértice V_i (origem) e V_j (destino)

Saída: 1 se aresta existe, 0 se aresta não existe ou -1 se grafo inconsistente

Pós-condição: nenhuma

Especificação do TAD Dígrafo

Operação ***remove_aresta:***

Entrada: o endereço do grafo e os identificadores do par de vértices (V_i e V_j)

Pré-condição: o grafo existir, os vértices serem válidos e a aresta desejada existir

Processo: remover a aresta existente entre os vértices V_i (origem) e V_j (destino)

Saída: 1 se sucesso, 0 se aresta não existe ou -1 se grafo inconsistente

Pós-condição: grafo de entrada com uma aresta a menos

Especificação do TAD Dígrafo

Operação ***consulta_aresta*:**

Entrada: o endereço do grafo, os identificadores do par de vértices (V_i e V_j), e o endereço da variável de retorno do peso da aresta

Pré-condição: o grafo existir, os vértices serem válidos e a aresta desejada existir

Processo: atribuir o peso da aresta existente entre os vértices V_i e V_j para a variável de retorno

Saída: 1 se sucesso, 0 se aresta não existe ou -1 se grafo inconsistente

Pós-condição: nenhuma

Especificação do TAD Dígrafo

Operação ***libera_grafo:***

Entrada: o endereço do endereço do grafo

Pré-condição: o grafo existir

Processo: liberar a área ocupada pelo grafo

Saída: nenhuma

Pós-condição: grafo inexistente

Especificação do TAD Dígrafo

Operação ***mostra_adjacentes*:**

Entrada: o endereço do grafo e o identificador de um vértice *V*

Pré-condição: o grafo existir e o vértice ser válido

Processo: apresentar o conjunto dos vértices adjacentes ao vértice *V*

Saída: nenhuma

Pós-condição: nenhuma

Especificação do TAD Dígrafo

Operação ***mostra_grafo:***

Entrada: o endereço do grafo

Pré-condição: o grafo existir

Processo: apresentar cada vértice do grafo e seu conjunto de vértices adjacentes

Saída: nenhuma

Pós-condição: nenhuma

Grafos: outras operações e algoritmos

Diversas outras operações podem ser definias de acordo com a necessidade de uma aplicação

Ex: *insere_vertice (G, V), remove_vertice (G, V), obtem_grau (G, V),* etc.

Algumas classes de algoritmos estudados em grafo:

- Percurso em um grafo (largura ou profundidade)
- Obtenção do menor caminho
- Geração da árvore geradora mínima
- Coloração de grafos
- Entre outros

Bibliografia

Slides adaptados do material da Profa. Dra. Denise Guliato.

CORMEN, T.H. et al. Algoritmos: Teoria e Prática, Campus, 2002

ZIVIANI, N. Projeto de algoritmos: com implementações em Pascal e C (2ª ed.), Thomson, 2004

MORAES, C.R. Estruturas de Dados e Algoritmos: uma abordagem didática (2ª ed.), Futura, 2003