Tipos Abstratos de Dados
TAD

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

Baseado nos slides do Prof. Rudinei Goularte

Objetivos

- ➔ Definir mais profundamente TADs
- ➔ Mostrar exemplos e implementações de TADs
- ➔ Apontar algumas técnicas mais avançadas para criar TADs

Algumas definições importantes

Algoritmos, Estruturas de Dados e Programas

- Algoritmo
  - Pode ser visto como uma sequência de ações executáveis* para a obtenção de uma solução para um determinado tipo de problema (Ziviani, 2003).

* Como sinônimo de factíveis

Algoritmos, Estruturas de Dados e Programas

- Estrutura de Dados
  - Organização de dados e operações (algoritmos) que podem ser aplicadas sobre os dados como forma de apoio à solução de problemas (complexos).
- Programas
  - Formulações concretas de algoritmos abstratos, baseados em representações e estruturas específicas de dados (Wirth, 1976) – algoritmos que podem ser executados em computadores

Algoritmos, Estruturas de Dados e Programas

- Tipo de Dado
  - Caracteriza o conjunto de valores a que uma constante pertence, ou que podem ser assumidos por uma variável ou expressão, ou que podem ser gerados por uma função (Wirth,1976).
    - Tipos simples: int, float, double, etc.
    - Tipos estruturados: structs

TAD

TAD

- Modelo matemático de tipos de dados
- Definido por seu comportamento (semântica) do ponto de vista do usuário dos dados
  - Especificamente em termos de valores possíveis, operações possíveis sobre o dado, e comportamento das operações
- Uma classe de objetos cujo comportamento lógico é definido por um conjunto de valores e operações

TAD

- É usado para encapsular tipos de dados (pensar em termos das operações suportadas e não como são implementadas)
  - Vantagem: organização!

TAD

- Separa o tipo de dado de sua representação
- Pode ser representado matematicamente por um par (v,o)
  - v = conjunto de valores
  - o = conjunto de operações sobre esses valores
- Ex: tipo real
  - v = R
  - o = {+, -, *, /, =, <, >, <=, >=}

TAD

- Requer que as operações sejam definidas sem estarem atreladas a uma representação específica dos dados
  - Ocultamento de informação
- Podemos usar listas, filas, pilhas, árvores, grafos, etc. sem precisar se preocupar com a implementação em si
  - E até mesmo com o tipo interno do valor de cada elemento, dependendo da implementação

TAD

- ➔ Não há necessidade de saber a representação interna de um tipo de dado
- ➔ Não se preocupa com a eficiência de tempo e espaço, porque elas são questões de implementação

Implementando um TAD

Implementando um TAD

- O conceito de TAD é suportado por algumas linguagens de programação procedimentais
  - Ex. Java, C, C++, Python ...

Implementando um TAD

- Para definir um TAD
  - O programador descreve o TAD em dois módulos separados
  - Um módulo contém a definição do TAD: representação (declaração) da estrutura de dados e implementação de cada operação suportada.
    - Em C, este módulo é um (ou alguns) arquivo .c

Implementando um TAD

- Para definir um TAD
  - O outro módulo contém a interface de acesso: apresenta as operações possíveis
    - Em C, este módulo é um arquivo .h
  - Outros programadores podem, por meio da interface de acesso, usar o TAD sem conhecer os detalhes representacionais e sem acessar o módulo de definição

Implementando um TAD

- Ocultação de informação (information hiding)
  - Os dados armazenados podem ser manipulados apenas pelas operações
  - Ocultamento dos detalhes de representação e implementação, sendo que apenas a funcionalidade é conhecida
  - Só se tem acesso às operações de manipulação dos dados, e não aos dados em si

Implementando um TAD

➔ Uma vez definido um TAD e especificadas as operações associadas, ele pode ser implementado em uma linguagem de programação

Implementando um TAD

- Uma estrutura de dados pode ser vista, então, como uma implementação de TAD
  - Implica na escolha de uma ED para representá-lo, a qual é acessada pelas operações que ele define
  - Uma ED é construída a partir dos tipos básicos (integer, real, char) ou dos tipos estruturados (array, struct) de uma linguagem de programação

Implementando um TAD

➔ Podem existir diversas implementações para um mesmo TAD, cada uma com suas vantagens e desvantagens

Vantagens do TAD

Vantagens do TAD

- Principais vantagens:
  - Reúso
  - Manutenção
  - Correção
  - Independência de representação

Exemplos

Exemplos

Dado do Mundo Real	Dados de interesse	Estrutura de armazenamento	Possíveis Operações
Pessoa	Idade	Tipo inteiro	➔ Nasce (i=0) ➔ Aniversário (i += 1)
Cadastro de Funcionário	Nome, cargo e salário de funcionários	Tipo lista ordenada	➔ Entra na lista ➔ Sai da lista ➔ Altera o cargo ➔ Altera o salário
Fila de espera	Nome e posição na fila de cada pessoa	Tipo fila	➔ Sai da fila (o primeiro) ➔ Entra na fila (no fim)
Baralho de cartas	Dados da carta e posição no baralho	Tipo pilha	➔ Entra na pilha (topo) ➔ Sai da pilha (topo)

Um TAD de Racionais

TAD Racionais

- A definição de valor para o TAD Racional,
  - Consiste em 2 inteiros, sendo o segundo deles diferente de zero
  - Dois inteiros que formam um número racional: numerador e denominador
- A operações do TAD Racional incluem:
  - Operações de criação, adição e multiplicação

TAD Racionais

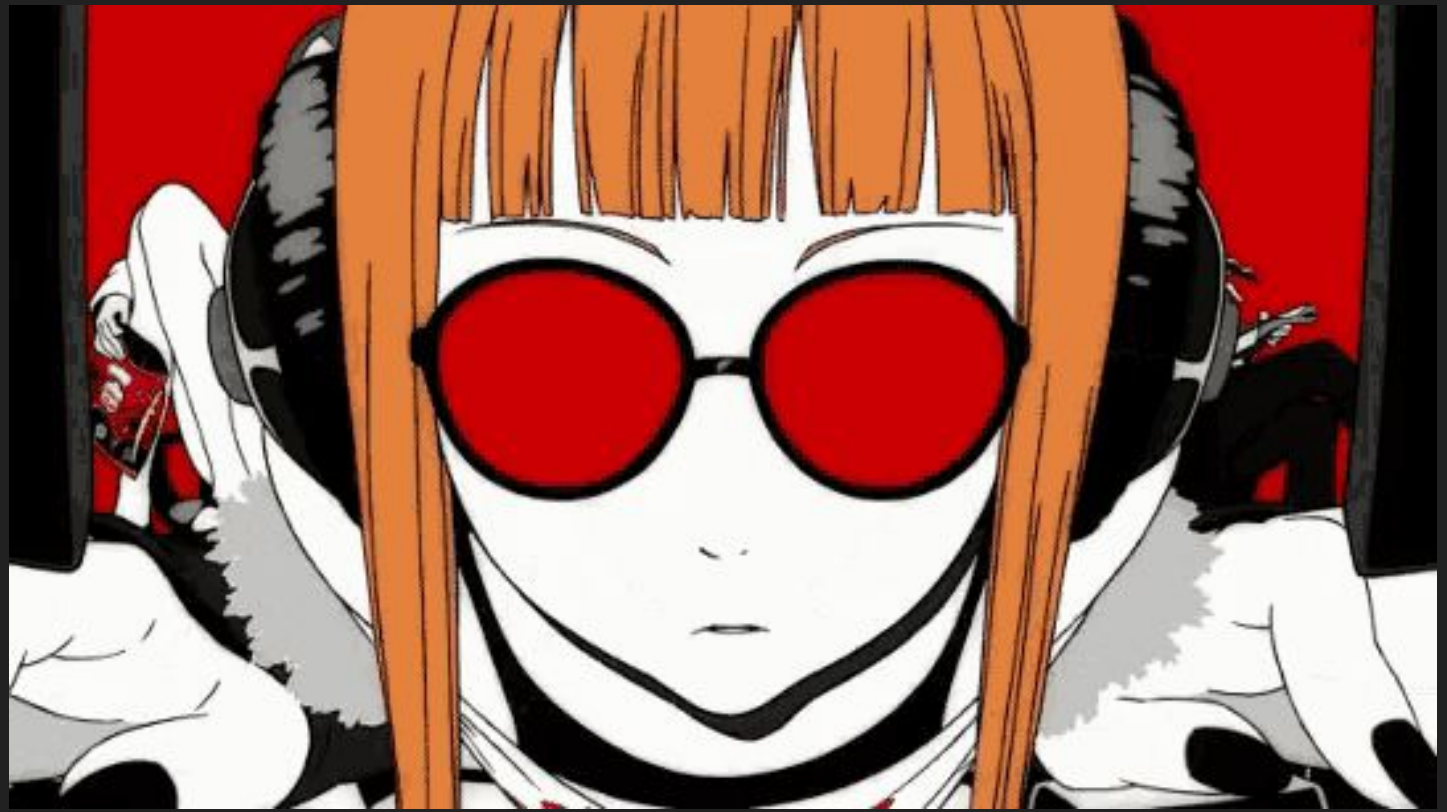
- Conceito matemático de um número racional
  - Pode ser expresso como o quociente de dois inteiros
  - As operações definidas são:
    - Criação de um número racional a partir de dois inteiros
    - Adição
    - Multiplicação

```
1 /* definição de valor */
2 Inteiro numerador;
3 Inteiro denominador;
4
5 /* definição de comportamentos */
6 Racional criar(Inteiro var1, Inteiro var2)
7 Pré-condição :
8    var2 != 0
9 Pós-condição :
10   numerador = var1
11   denominador = var2
12
13 Racional adição(Racional var1, Racional var2)
14 Pré-condição :
15   nenhuma
16 Pós-condição :
17   numerador = (var1.numerador * var2.denominador) + (var2.numerador * var1.denominador)
18   denominador = var1.denominador * var2.denominador
19
20 Racional multiplicação(Racional var1, Racional var2)
21 Pré-condição :
22   nenhuma
23 Pós-condição :
24   numerador = var1.numerador * var2.numerador
25   denominador = var1.denominador * var2.denominador
```

TAD Racionais

- Implementar significa mapear a estrutura de dados e as operações em uma linguagem de programação (que o computador entenda)
  - Neste curso, a empregada será C

Vamos Programar!



Considerações Finais

- ➔ Na implementação de um TAD, a escolha da estrutura de dados empregada tem papel importante
  - ◆ Uma escolha mal feita pode resultar em implementações ineficientes ou mesmo não-factíveis

Referências

- ZIVIANI, N. Projeto de Algoritmos, Thomson, 2a. Edição, 2004.