SCC0202 – Algoritmos e Estruturas de Dados I

Tipos Abstratos de Dados

Prof.: Dr. Rudinei Goularte

(rudinei@icmc.usp.br)

Instituto de Ciências Matemáticas e de Computação - ICMC

Sala 4-229

Conteúdo

- Definição de Tipos Abstratos de Dados (TADs)
- Exemplo de um TAD
- Implementação

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

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).
- 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

* Como sinônimo de factíveis

Definição de TAD

- Um “tipo abstrato de dados” é um conjunto de valores e uma sequência de operações sobre estes valores
- Este conjunto e estas operações formam uma construção matemática que pode ser implementada usando determinada estrutura de dados do hardware ou do software
- “Tipo abstrato de dados” refere-se ao conceito matemático básico que define o tipo de dado
  - É usado para encapsular tipos de dados (pensar em termos das operações suportadas e não como são implementadas)
    - Vantagem: organização!
  - 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

(Tenenbaum, 2004)

Definição de TAD

- Modelo Matemático
  - Um TAD pode ser visto como uma tupla (*v*,*o*), onde
    - v é o conjunto de valores
    - o é o conjunto de operações aplicadas sobre esses valores
  - Exemplo, TAD Números Reais
    - v = R
    - o = {+,-,*,/,<,>,=,<=,>=}

Ocultamento de Informação

- Os dados armazenados podem ser manipulados apenas pelas operações definidas na interface – *Information Hiding* (Ocultamento de Informação)
  - 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 !!!!**

TAD e Estruturas de Dados

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

Vantagens do Uso de TAD

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

  - Exemplo “abstrato” :-)
    - Conversão decimal-binário
      - Array x Pilha

Como implementar um TAD?

- Requer que operações sejam definidas sobre os dados sem estarem atreladas a uma representação específica
  - programador que usa um tipo de dado real, integer, array, não precisa saber como tais valores são representados internamente
  - O mesmo princípio pode ser aplicado a listas, pilhas, ...
  - Se existe uma implementação disponível de uma lista, p. ex., um programador pode utilizá-la como se fosse uma caixa preta, e acessá-la por meio das operações que ela suporta

Como implementar um TAD?

- O conceito de TAD é suportado por algumas linguagens de programação procedimentais
  - Ex. Java, C, C++, Python ...
- Para definir um TAD
  - O programador **projetista** 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.

    [Nota: em C, este módulo é um (ou alguns) arquivo .c]
  - O outro módulo contém a interface de acesso: apresenta as operações possíveis

    [Nota: em C, este módulo é um arquivo .h]
  - Os programadores **usuários** podem, por meio da interface de acesso, usar o TAD sem conhecer os detalhes representacionais e sem acessar o módulo de definição

Exemplo de um TAD

- 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

Exemplo de um TAD

- TAD Racional
  - 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

Exemplo de um TAD

```
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
```

Implementando o TAD

- 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**

Implementando o TAD

```
1  #ifndef RACIONAL_H
2     #define RACIONAL_H
3
4     typedef struct racional RACIONAL;
5
6     RACIONAL* criar(int num, int den);
7     void apagar(RACIONAL* rac);
8
9     RACIONAL* adicao(RACIONAL* v1, RACIONAL* v2);
10    RACIONAL* multiplicacao(RACIONAL* v1, RACIONAL* v2);
11
12    void imprimir(RACIONAL* rac);
17
18 #endif
/*arquivo racional.h*/
```

Implementando o TAD

```
1 #include "racional.h"
2
3  struct racional{
4    int num;
5    int den;
6  };
7
8  RACIONAL* criar(int num, int den) {
9    ...
10 }
11
12 void apagar(RACIONAL* rac) {
13   ...
10 }
15
16 void imprimir(RACIONAL* rac) {
17   ...
18 }
19
20 RACIONAL* adicao(RACIONAL* v1, RACIONAL* v2) {
21   ...
22 }
23
24 RACIONAL* multiplicacao(RACIONAL* v1, RACIONAL* v2) {
25   ...
26 }
/*Arquivo racional.c*/
```

Implementando o TAD

```
1 #include "racional.h"
2
3 int main() {
4    RACIONAL* r1 = criar(1, 2);
5    RACIONAL* r2 = criar(1, 3);
6    imprimir(r1);
7    imprimir(r2);
8
9    RACIONAL* r3 = adicao(r1, r2);
10   imprimir(r3);
11
12   RACIONAL* r4 = multiplicacao(r1, r2);
13   imprimir(r4);
14
15   apagar(r1);
16   apagar(r2);
17   apagar(r3);
18   apagar(r4);
19
20   return 0;
21 }
```

Implementando o TAD

- Makefile

```
1 all: racional.o main.o
2    gcc main.o racional.o -o racional -std=c99 -pedantic-errors -Wall –lm
3
4 racional.o:
5     gcc -c racional.c
6
7 main.o:
8     gcc -c main.c
9
10 clean:
11    rm *.o racional
```

- Opções
  - -std: indica o padrão C a ser seguido na compilação
  - -o: define nome de arquivo de saída
  - -Wall: mostra todos os warnings
  - -pedantic-errors: mostra todos os erros, independente de padrão (std)
  - -lm: inclui biblioteca matemática
  - -c: somente compila (gera *.o)

Implementação

- 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

- Fim da Aula.