



Universidade Federal do Vale do São Francisco

Algoritmos e Estruturas de Dados I

Professor: Marcelo Santos Linder

E-mail: marcelo.linder@univasf.edu.br

Ementa

- Noções de abstração de dados.
- Vetores e matrizes.
- Pilhas, filas, listas, árvores binárias, árvores AVL e árvores 234: alocação estática e dinâmica e algoritmos de inserção, remoção e consulta. Algoritmos recursivos.

Forma de Avaliação

- A avaliação será realizada mediante a aplicação de duas provas e um trabalho. A média do aluno na disciplina será calculada através do computo da média aritmética obtida com base nas notas do discente.

OBS.: O trabalho poderá vir a ser substituído por uma prova.

Informações Gerais

- Material de apoio
 - Os slides utilizados em aula, descrição de trabalhos, datas de avaliações e demais informações referentes à disciplina encontram-se na página (Ex.: PD e Planej.)

www.univasf.edu.br/~marcelo.linder

Introdução

Programa

Um programa pode ser visto como a especificação formal da solução de um problema. N.Wirth expressa em sua equação

programa = algoritmo + estruturas de dados

onde: o algoritmo contém a lógica do programa e os dados são organizados em estruturas de dados.

Estruturas de Dados

A qualidade da solução de um problema depende, entre outros fatores, da forma como estão organizados os dados relevantes.

- Encontrar o número do telefone de um certo assinante no conjunto de todos os assinantes.
- Representação interna de uma string:

- | | | | | |
|---|---|---|---|---|
| 4 | G | A | T | O |
|---|---|---|---|---|

;
- | | | | | |
|---|---|---|---|----|
| G | A | T | O | \0 |
|---|---|---|---|----|

.

Toda uma classe de modelos desenvolveu-se, ao longo do tempo, com o objetivo de viabilizar o processamento de dados.

Estruturas de Dados

Estruturas de dados são formas genéricas de se estruturar informação de modo a serem registradas e processadas pelo computador.

Ex.:

- ✦ vetores;
- ✦ lista ordenada;
- ✦ árvores;
- ✦ grafos, etc.

Contudo, estas só adquirem significado quando associadas a um conjunto de **operações**, que visam, de um modo geral, manipulá-las (algoritmos).

Tipos Abstratos de Dados

Embora os termos “tipo de dado”, “estrutura de dados” e “tipo abstrato de dados” em essência se refiram aos mesmos objetos formais, eles são usados em contextos próprios.

O termo “tipo de dado” é usado no contexto de uma linguagem de programação (tipos primitivos) e está associado a um método de interpretar um padrão de bits.

Tipos Abstratos de Dados

O termo “tipo abstrato de dado” (TAD) denota um modelo junto com um conjunto de operações definidas sobre o modelo. (tipo + operações válidas).

Por fim, “estrutura de dados” é uma forma concreta de se implementar um TAD, ou seja, uma representação computacional do modelo matemático em questão.

Tipos Abstratos de Dados

Para viabilizar a implementação de tipos abstratos de dados vamos nos valer das estruturas em C. Para exemplificar este conceito, definiremos um TAD RACIONAL.

Um número racional pode ser expresso como o quociente de dois inteiros. Definiremos a operação de criação e multiplicação de números racionais.

Tipos Abstratos de Dados

```
typedef struct
{
    int num;
    int den;
}RACIONAL;
void criar_racional (int, int, RACIONAL *);
void multiplicar_racionais (RACIONAL *,
    RACIONAL *, RACIONAL *);
```

Tipos Abstratos de Dados

```
void criar_racional (int n, int d, RACIONAL *r)
```

```
{
```

```
    r->num = n;
```

```
    (*r).den = d;
```

```
}
```

```
void multiplicar_racionais (RACIONAL *a,  
    RACIONAL *b, RACIONAL *c)
```

```
{
```

```
    c->num = a->num * b->num;
```

```
    c->den = a->den * b->den;
```

```
}
```

Tipos Abstratos de Dados - Exercício

Implemente as operações de soma e verificação de equivalência entre elementos do TAD RACIONAL, definido anteriormente.

Tipos Abstratos de Dados - Exercício

```
void somar_rationais (RACIONAL *a, RACIONAL *b,  
    RACIONAL *c)  
{  
    c->num = a->num * b->den + b->num * a->den;  
    c->den = a->den * b->den;  
}  
  
int equivalencia_rationais (RACIONAL *a,  
    RACIONAL *b)  
{  
    return a->num * b->den == a->den * b->num;  
}
```