Estatística - Listas de Exercícios

Jorge Augusto Salgado Salhani

Agosto, 2022

1 Lista 1

1.1 O que é um Tipo Abstrato de Dados (TAD) e qual a característica fundamental na sua utilização?

Em primeiro lugar, definimos um "tipo de dado" como um conjunto de valores que uma dada variável assume. Por exemplo,

```
int x = 10;
double d = 5.7;
```

são tipos de dados simples. Já a parte "abstrata" está em considerar a representação de um problema em particular. Em exemplo, temos a representação de números complexos. Um número complexo é representado por dois valores reais (tipo simples **int** ou **double**) que, combinados, na forma

```
typedef struct Complex{
    int im;
    int re;
} Complex;
```

geram uma abstração chamada **Complex**.

Dessa forma, um tipo abstrato de dado (TAD) carrega dois elementos principais. São eles:

1. são definidas as operações possíveis sobre o dado definido
2. são ocultadas tanto as informações de implementação quanto o acesso aos dados em si (ou **information hiding**)

De modo conciso, TAD (Tipo Abstrato de Dado) é um tipo de dado cujas propriedades (domínio - valores possíveis - e operações) são especificadas independentemente da implementação.

1.2 Quais as vantagens de se programar com TADs?

1. Segurança de uso: garante que apenas operações pré definidas sejam executadas sobre um tipo específico de dados, sem que haja manipulação direta da estrutura dos dados armazenados.
2. Modularização: permite manutenção do código e implementação de novas operações, além de garantir melhor organização dos processos implementados (distinto de códigos "monolíticos")
3. Interface de acesso: dispõe ao usuário final o conjunto completo de operações disponíveis para o TAD (Tipo Abstrato de Dado) implementado, permitindo o uso da estrutura de dados como uma "caixa preta", sem que se conheça a implementação
4. Reutilização: permite o uso da mesma estrutura independentemente da implementação do usuário

1.3 Considere dois programas envolvendo o cadastro de funcionários. O programa A foi construído de acordo com os princípios de TAD. Já o programa B, não. Diferencie os dois programas.

Podemos distinguir os programas de forma situacional, ou seja, conforme alguns contextos de uso. Em cada programa, faremos as operações de:

1. Cadastrar novo funcionário (create)
2. Resgatar informações (read / get)
3. Atualizar informações (update)
4. Deletar cadastro (delete)

Para o cadastro de novo funcionário, o usuário do programa A deverá ter acesso aos parâmetros necessários para se cadastrar um funcionário (e.g. nome, idade, cargo) e imputá-los em uma função responsável pela inserção de um novo item (funcionário).

```
createNewEmployee(name, age, role);
```

Não é necessário que o usuário se preocupe com o modo como a adição de um novo cadastro acontece.

Já para o usuário do programa B, é necessário que conheça tanto os parâmetros de cadastro (similar ao programa A) quanto a forma como os dados são armazenados, para que em seguida faça sua inserção na base de dados.

Por exemplo, caso a base de dados de funcionários tenha sido implementada utilizando um array simples, o cadastro será distinto se caso tenha sido implementada utilizando dicionário (relação chave:valor).

Para resgatar / ler informações sobre um ou mais funcionários já cadastrados, o usuário do programa A deverá ter acesso aos parâmetros que permitem a busca. Por exemplo, podemos ter uma busca por nome

```
getEmployeeByName(name, maxResult);
```

Que é responsável por retornar uma lista de resultados (definido por construção) contendo os itens de cadastro.

Já para o usuário do programa B, de modo análogo à execução anterior, deve se ter acesso à forma como os dados estão armazenados na base de dados e fica a critério do usuário o mecanismo de busca por nome de funcionários já cadastrados.

Para atualização e deleção de cadastros, as diferenças são análogas, deixando evidente as vantagens mencionadas no exercício anterior. O programa A garante segurança dos dados por meio da ocultação de informação; reutilização de métodos implementados e disponibilizados em interface de acesso; e manutenção do código, devido à modularidade da implementação.

1.4 Faça a especificação de um sistema de controle de empréstimos de uma biblioteca usando TAD: diga quais os dados e as operações, e especifique como organizar os dados e operações durante a implementação.

Assim como descrito antes, as ações básicas esperadas para o gerenciamento do controle de empréstimos são ações de novo cadastramento (create), resgate de informações (read/get), atualização de cadastro (update) e descadastramento (delete). De modo geral utilizaremos os métodos CRUD (create, read, update, delete) como estrutura básica de implementação.

É esperado que clientes possam solicitar empréstimos e realizar agendamentos de empréstimos. Para isso, precisaremos de duas TADs distintas. Uma estante contendo itens BOOK e uma base de solicitações contendo itens SCHEDULING

Um item BOOK será composto dos seguintes dados

```
/* book.h */
#include "scheduling.h"
typedef struct book_ BOOK;

// CREATE
BOOK* create_book(char* title, char* author);
```

```
// READ
BOOK* get_book_by_id(int id);
int get_book_id(char* title, char* author);
int get_available_amount(int id);
int get_reservation_amount(int id);
int get_scheduling_amount(int id);
SCHEDULING* get_scheduling(int id);
void printf_book(int id);

// UPDATE
bool set_title(int id, char* title);
bool set_author(int id, char* author);
bool set_available_amount(int id, int available_amount);
BOOK* borrow_book(char* title, char* author, char* requester);

// DELETE
bool delete_book(int id);

/* livro.c */
struct book_ {
    int id;
    char* title;
    char* author;
    int available_amount;
    SCHEDULING* scheduling;
};
```

Um item SCHEDULING será composto dos seguintes dados

```
/* scheduling.h */
typedef struct scheduling_ SCHEDULING;

// CREATE
SCHEDULING* new_scheduling(char* requester, int amount);

// READ
SCHEDULING** get_scheduling_by_book_id(int id);

/* scheduling.c */
struct scheduling_ {
    int id;
```

```
        char* requester;
        int amount;
    }
```

A forma de armazenamento de cada item BOOK deve ser construída como um *array* ordenado alfabeticamente pela propriedade *title*, pra que seja feita buscas binárias para resgatar o id do livro.