MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto
Departamento de Computação e Sistemas
Campus João Monlevade

TIPOS ABSTRADOS DE DADOS: CONTEXTO E APLICAÇÃO

Curso: Sistemas de Informação — 1º semestre de 2020
Disciplina: Algoritmos e Estruturas de Dados I — Data: 06/03/20
Professor: Alexandre Magno de Sousa

1 Contexto

Uma equipe de desenvolvedores está projetando um novo Website para redes sociais. Para tanto, eles precisam amazenar os seguintes dados do perfil de usuários: código de identificação do usuário (`ID`), nome, email, ano de nascimento, login e senha, lista de amigos. O `ID` nada mais é do que o índice do vetor em que o usuário foi cadastrado no arranjo do TAD.

Quando um usuário envia um convite de amizade para outro usuário e esse convite é aceito, os dois tornam-se amigos na rede social. Os analistas estudam uma forma de armazenar a lista de amigos de um usuário, pois criar um vetor como atributo da estrutura de dados para armazenar a lista de amigos consome muita memória. Na verdade consome duas vezes mais memória, pois se um usuário A é amigo de um usuário B, a lista de amigos do usuário A armazenará o usuário B e a lista de amigos do usuário B armazenará o usuário A.

Para expressar a rede de amizades da rede social uma matriz $n \times n$ de inteiros é criada para armazenar todos os relacionamentos de amizade da rede, onde n representa o númerto total de usuários cadastrados na rede. Inicialmente, essa matriz é inicializada com zeros. Assim, se um usuário A, representado pela i-ésima linha da matriz, é amigo de um usuário B, representado pela j-ésima coluna, a matriz na posição (i, j) receberá o valor 1, indicando que o usuário A é amigo do usuário B. Da mesma forma, como o relacionamento de amizade é bidirecional a matriz de posição (j, i) também recebe 1. Lembrando que a diagonal principal desse matriz será sempre 0, pois um usuário nunca poderá ser amigo dele mesmo.

2 Arquitetura de Software

O ciclo de vida tradicional do desenvolvimento de software envolve as fases de análise, projeto, implementação, teste e manutenção. Na fase de análise, determinamos o que o software precisa fazer, quais problemas deverá resolver e quais informações deverá manipular.

Na fase de projeto, o objetivo é definir como o software será desenvolvido. Estamos falando de uma definição geral, não detalhada. Na fase de projeto, escolhemos a arquitetura do software, definimos quais são os principais módulos e, com isso, podemos dividir o trabalho entre os desenvolvedores - cada membro da equipe fica responsável pela implementação de um dos módulos. A Figura 1 apresenta uma visualização simples da arquitetura do software na qual o software é construído sobre duas bases: (i) os Tipos Abstratos de Dados (TAD) com seus tipos definidos e operações sobre esses tipos, os módulos de acordo com o contexto do problema; e, por fim, (ii) interface gráfica com o usuário, que é o meio visual de comunicação entre o usuário do sistema e os objetos do software.

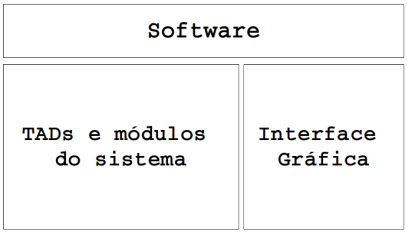

Figura 1: Arquitetura do Software.

As outras fases, referem-se à implementação (codificação em uma linguagem de programação), teste (para garantir que o software funcione segundo o esperado) e manutenção (ajuste ou evolução nas funcionalidades do software, ao longo de seu perído de utilização). Para uma leitura complementar sobre o

ciclo de vida tradicional do desenvolvimento de software, consulte Sommerville[1] e Pressman[2].

Diante disso, independente do problema a ser solucionado pelo software que será desenvolvido, é extremamente importante definir qual será a arquiteura do sitema na fase de projeto.

3 Tipo Abstrato de Dados (T.A.D.)

Um TAD envolve dois elementos essenciais: (a) os dados e (b) as operações sobre esses dados. Os dados são obtidos a partir da abstração do problema real, representam as informações necessárias envolvidas no processo do problema. O recurso da linguagem C utilizado para a construção dos tipos de dados necessários são os registros ou `structs`, também são chamados de tipos de dados heterogêneos.

As operações nada mais são do que "aquilo que os sitema pode fazer" com os dados e, geralmente, o nome de uma operação é representado por um verbo no infinitivo pois indica uma ação realizada sobre os dados (e.g. cadastrar, alterar, excluir ou pesquisar). Além disso, as operações deverão apresentar nomes intuitivos que representam a responsabilidade da operação. Em uma linguagem de programação, as operações são representadas por meio de funções ou procedimentos[3]. Na linguagem C, funções e procedimentos são representados por meio de um bloco chamado de função.

Diante disso, são declarados os seguintes tipos para os dados do problema:

```
typedef struct data{
 int dia;
 int mes;
 int ano;
}TData;

typedef struct usuarios{
 char nome[100];
 char email[200];
 char login[50];
 char senha[50];
 TData data_de_nascimento;
 int ID; // deverá ser gerado pelo programa e
         // igual ao índice do elemento cadastrado no vetor (lista)
}TUsuarios;

typedef struct redeSocial{
 TUsuarios vetor[100];// lista de usuários
 int matrix[100][100]; // matriz de relacionamentos de amizade
 int indice; // indica a posição em que o próximo elemento vai ser inserido na lista e
             // também informa a quantidade de usuários cadastrados na lista
}TRedeSocial;

// Declaração das variáveis dentro da função main()
TUsuario user;
TRedeSocial rede;
```

As operações deverão ser implementadas tendo como base a divisão de responsabilidades, ou seja, cada função deverá fazer apenas o que é de sua responsabilidade e não deverá apresentar código repetido de outra função, quando for necessário, o uso de funções já implementadas deverá ser feito, baseado no reaproveitamento de código.

Para construção das operações, sempre deve-se pensar primeiro:

[1] Sommerville, R. I. **Engenharia de Software**. 6 ed. São Paulo: Addison Wesley, 2003. p. 35-38

[2] Pressman, R. **Engenharia de Software**. 3 ed. São Paulo: Makron Books, 1995. p. 32-35

[3] Funções e procedimentos podem receber como entrada parâmetros com os quais eles podem trabalhar, esse processo é chamado de passagem de parâmetro (que pode ser por valor ou referência). A principal diferença entre função e procedimento é que uma função sempre retorna algum valor, enquanto que o procedimento não retorna um valor depois de realizar a sua operação.

1. O que você pode fazer com um elemento (objeto) `TUsuario`? A partir daqui, as operações básicas podem ser desenvolvidas;

2. O que cada operação precisa, em termos de dados, para ser realizada? Ou seja, quais parâmetros são necessários? Deve-se utilizar passagem de parâmetro por valor ou por referência?

Observação: não serão declaradas variáveis globais para o desenvolvimento do programa, pois o uso de variáveis globais não é uma boa prática de programação!

3.1 Operações Auxiliares

- `Ler`: recebe como passagem de parâmetro por referência um `TUsuario` e realiza a leitura de todos os dados de um usuário. **Observação:** para leitura de strings a função `fgets` será obrigatória, o uso da função `fflush(stdin)` deverá ser feito antes de qualquer função de leitura, seja `fgets` ou `fflush`, isso evita que o programa salte leituras durante a execuação.

- `Imprimir`: recebe como passagem de parâmetro por valor um `TUsuario` e realiza a impressão de todos os dados da estrutura.

3.2 Operações Básicas

- `Iniciar`: inicializa o campo `indice` da rede social (`TRedeSocial`) e inicializa a matriz de relacionamentos de amizades com zeros;

- `Cadastrar`: cadastra um elemento (`TUsuario`) na estrutura de dados (`TRedeSocial`);

- `Imprimir 2`: realiza a varredura no vetor (lista) de elementos (`TUsuario`) da rede social (`TRedeSocial`) e imprime todas as informações de um elemento;

- `Pesquisar`: realiza a pesquisa de um `ID` de um elemento (`TUsuario`) passado como parâmetro no vetor de elementos da rede social (`TRedeSocial`) e retorna o índice de onde esse elemento (`TUsuario`) se encontra;

- `Alterar`: recebe como parâmetro o índice de um elemento já pesquisado para realizar a alteração dos dados e recebe um elemento (`TUsuario`) para que seja atualizado, assume-se que esse elemento já vem alterado, ou seja, antes da chamada da função `Alterar` deve-se fazer o uso da função `Pesquisar` e da função auxiliar `Ler`, essa realiza a leitura dos novos dados do usuário;

- `Excluir`: recebe como parâmetro o índice de um elemento já pesquisado para realizar a exclusão dos dados e, neste caso, para excluir um usuário basta que o `ID` receba -1. Não deverá ser realizado o deslocamento dos elementos dentro do vetor (lista) da rede social (`TRedeSocial`) para rearranjá-lo, pois cada elemento possui como `ID` o índice de onde ele foi cadastrado. Não se esqueça de atualizar a matriz de relacionamentos de amizade, pois, se um usuário é excluído, os seus amigos não terão mais um relacionamento de amizade com ele. Antes da chamada da função `Excluir` deve-se fazer o uso da função `Pesquisar` para identificar o índice do elemento (`TUsuario`) a ser excluído.

Observação: lembre-se de se perguntar do que cada função precisa para realizar a operação, ou seja, quais dados são necessários para realizar a operação, e quais deles devem ou não ser por passagem de parâmetro por valor ou referência.

4 Modularização e Interface Gráfica com o Usuário

A Interface Gráfica com o usuário representa um papel fundamental na arquitetura de software do sistema, pois é por meio dela que o usuário do sistema poderá interagir com o sistema e fornecer entradas e também é por meio dela que o sistema poderá fornecer saídas ao usuário do sistema.

Para este projeto, a Interface Gráfica será construída por meio da Opção de Menus. A Opção de Menus dão ao usuário a opção de realizar alguma operação básica no sistema. A operação básica escolhida pelo usuário do sistema geralmente representa uma função de operação básica do TAD: `Inserir`, `Pesquisar`, `Imprimir`, `Alterar` e `Excluir`.

Nesse contexto, a Modularização de Menus significa "dividir os menus em módulos", nesse caso, os módulos são funções, ou seja, devemos dividir os menus em várias funções.

Se um programa trabalha com várias estruturas de dados diferentes para cada tipo de objeto, o ideal é separar o programa em módulos. Além disso, deverá existir um Menu principal onde o usuário terá a opção de escolher com qual módulo deverá trabalhar. Assim, cada módulo será responsável por um elemento do sistema (por exemplo: usuário ou amizade – matriz de relacionamento que existe no elemento `TUsuarios`) e cada um deverá apresentar seu próprio Submenu de opções.

Para construção de Menus e Submenus serão utilizados os seguintes recursos da Linguagem C:

(a) Comando `switch-case`;

(b) Laço de repetição `do-while`.

5 Módulo de Gerenciamento de Amizades

Esse módulo deverá ser implementado para que o sistema consiga realizar o cadastro de amizades entre dois ou mais usuários da rede social. Portanto, funções deverão ser projetadas de acordo com as operações básicas desse módulo as quais estão descritas a seguir. Além disso, deverá ser necessário desenvolver uma função de submenu específica para esse módulo na biblioteca de interface gráfica com o usuário.

Operações Básicas:

1. `Cadastrar Amizades`: deve receber como parâmetros dois nomes de usuários A e B (`TUsuarios`), pesquisá-los no cadastro de usuários da rede social (lista de usuários). E, com os índices i e j dos usuários A e B, respectivamente, cadastrar na matriz de relacionamentos de acordo com o índice de cada um desses dois usuários o relacionamento de amizade bidirecional, ou seja:
   - (a) do ponto de vista de que A no índice i é amigo de B no índice j atribuindo na linha i e coluna j da matriz, posição (i, j), o valor "1";
   - (b) do ponto de vista que B no índice i é amigo de A no índice j atribuindo na linha j coluna i da matriz, posição (j, i), o valor "1";
2. `Pesquisar Amizades`: Essa função deverá receber como parâmetro dois nomes de usuários A e B (`TUsuarios`) e pesquisar no vetor de usuários (lista de usuários) da rede social até encontrar o usuário A, obtendo seu índice i, e, também pesquisar no vetor de usuários da rede social até encontrar o usuário B, obtendo seu índice j. Com esses dois índices, verificar nas posições (i, j) e (j, i) da matriz se existe o valor "1", caso isso seja verdade, a função `Pesquisar Amizades` deverá retornar "1" (verdadeiro), caso contrário, a função deverá retornar "0" (falso);
3. `Excluir Amizades`: deve receber como parâmetros dois nomes de usuários A e B (`TUsuarios`), pesquisá-los no cadastro de usuários da rede social (lista de usuários). Da mesma forma com o que foi feito na função `Cadastrar Amizades`, por meio dos índices i e j dos usuários A e B, respectivamente, atribuir o valor "0" para posição da matriz (i, j) e para a posição da matriz (j, i), pois o relacionamento de amizades é bidirecional;
4. `Imprimir a Lista de Amigos`: essa função deve imprimir os dados dos usuários que são amigos de um determinado usuário passado como parâmetro. Para isso, a função deverá procurar por esse usuário na lista de usuários do sistema. Caso esse usuário passado como parâmetro esteja de fato cadastrado, a função deverá fazer uma varredura na matriz de relacionamentos na linha da matriz referente a esse usuário e, para cada posição da matriz em que houver o valor "1", a função deverá imprimir todos os dados desse usuário amigo do usuário passado como parâmetro por meio da primeira versão da função `Imprimir`.

6 Visão Geral da Modularização do Projeto

A Figura 2 apresenta uma visão geral da modularização ou divisão de bibliotecas do projeto do TAD `Rede Social`. A seguir são descritas cada um das bibliotecas e suas principais funcionalidades:

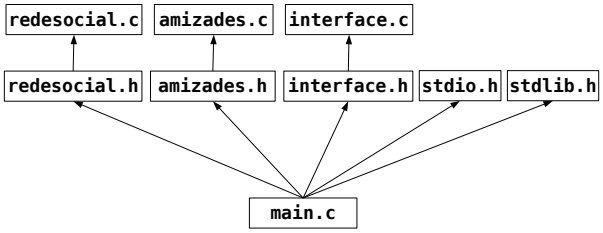

Figura 2: Diagrama das Bibliotecas do Projeto.

(a) `redesocial.h`: responsável pelo TAD `TRedeSocial` que contém elementos do tipo `TUsuarios` e operações auxiliares e básicas para realizar o gerenciamento de usuários;

(b) `amizades.h`: responsável pelo TAD `Amizades` que a matriz de relacionamentos como um elemento/objeto, também é onde se encontram as operações básicas para o gerenciamento de amizades do usuário;

(c) `interface.h`: responsável pelo módulo de Interface Gráfica do Usuário e contém operações com `Menu` e `Submenus`.

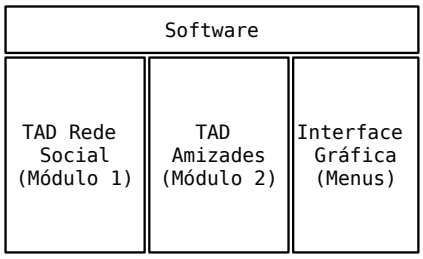

Figura 3: Arquitetura do Software.

Na Figura 3 é apresentada a arquitetura de software final contendo todos os módulos que são definidos pelas bibliotecas `redesocial.h`, `amizades.h` e `interface.h`. Todas essas bibliotecas são controladas por meio do arquivo `main.c` o qual contém a função principal `int main( )` que inicia a execução do sistema de redes sociais.

7 Exercícios para desenvolvimento de funções

Construa as funções a seguir:

(a) Uma função que encontre o usuário da rede social que tenha o maior número de amigos e retorne o `ID` do usuário.

(b) Uma função que recebe como parâmetro dois `ID`s de usuários da rede social, que não tem um relacionamento de amizade, mas que retorna a quantidade de amigos em comum que esses dois usuários tenham.

(c) Uma função que calcula uma matriz de amigos em comum $n \times n$ tal que o valor da i-ésima linha e j-ésima coluna representa o número de amigos em comum entre o usuário i e j. Essa matriz deve ser declarada dentro do módulo da rede social, para isso, altere a estrutura do módulo.

(d) Uma função que encontre dois usuários da rede social que não tenham nenhum amigo em comum e registre-os no módulo da rede social. Para isso, realize as modificações necessárias na estrutura de dados da rede social. Além disso, diga se a função deve ser passagem de parâmetro por valor ou por referência.

(e) Uma relação de amizade indireta acontece quando dois usuários A e C não são amigos um do outro, mas possuem um amigo em comum tal que A é amigo de B e B é amigo de C, dessa forma, uma amizade indireta é descrita como: $A \longleftrightarrow B \longleftrightarrow C$. Esta relação de amizade indireta é considerada de 1 nível, pois possui apenas um usuário entre A e C. Faça uma operação que encontre uma relação indireta de 1 nível entre dois usuários (e.g. i e j) da rede social e retorna o `ID` do usuário intermediário.

(f) Faça uma operação que encontre uma relação indireta de 2 (dois) níveis entre dois usuários (e.g. i e j) da rede social e que retorne, via passagem de parâmetro por referência, os dois `ID`s dos usuários intermediários.