Trabalho Prático de Estrutura de Dados I

Prazo para envio: 09/03/2025, 23:59, pelo AVA

1 Descrição do Problema

Vianne Rochele é uma mulher empreendedora determinada a transformar sua paixão por gastronomia, cultura e entretenimento em um negócio inovador. Decidida a abrir sua própria loja de varejo online, ela estrutura sua plataforma em quatro departamentos principais: chocolates, vinhos, livros e streaming de filmes. Para garantir um gerenciamento eficiente dos produtos disponíveis em cada categoria, Vianne precisa que o sistema seja implementado em uma estrutura dinâmica que permita adicionar, remover e reorganizar os itens de forma ágil, além de possibilitar a reversão de alterações quando necessário.

O sistema precisa ser capaz de gerenciar os produtos de cada departamento de maneira independente. Isso significa que o gerenciamento dos filmes deve ser totalmente isolado das demais categorias, como livros e chocolates, evitando qualquer tipo de interdependência que possa comprometer a eficiência ou causar erros operacionais.

A flexibilidade do sistema é outro fator crucial. Vianne precisa de uma plataforma que permita ajustes rápidos, como a adição de novos produtos ou a remoção de itens descontinuados, com um processo simples de controle de versão que garanta a segurança de cada alteração realizada. Isso assegura que a loja possa evoluir conforme as necessidades do mercado e as preferências dos consumidores, sem comprometer a integridade do estoque ou a qualidade do serviço prestado.

Para garantir uma gestão eficaz dos produtos em cada categoria, o sistema deve ser capaz de registrar informações detalhadas e específicas de cada tipo de item. Esses dados não só são essenciais para o controle de qualidade, mas também para proporcionar aos clientes uma experiência de compra mais rica e detalhada. Além disso, a estrutura de dados utilizada deve ser capaz de lidar com o volume de informações de maneira eficiente, com facilidade para atualizações, exclusões e adições de dados.

Os dados que devem ser gerenciados pelo sistema para cada tipo de produto incluem:

- Chocolate:
  - Nome (chave de busca)
  - Marca
  - Tipo de chocolate (ao leite, amargo ou branco)

  - Porcentagem de cacau (para chocolates amargos)
  - Nacionalidade ou origem
  - Peso
  - Data de fabricação
  - Validade

- Vinho:
  - Nome (chave de busca)
  - Vinícola
  - Tipo (tinto, branco, rosé, espumante branco, espumante rosé)
  - País
  - Uva
  - Região
  - Ano de fabricação

- Livro:
  - Título (chave de busca)
  - Autor
  - Editora
  - Data de publicação
  - Idioma
  - Número de páginas

- Filme:
  - Título (chave de busca)
  - Diretor
  - Gênero
  - Distribuidor
  - Duração
  - País de origem

Com base nesses requisitos, Vianne contratou sua empresa para desenvolver o sistema de gerenciamento das listas de produtos. Seu supervisor imediato lhe solicitou que a estrutura de dados fosse implementada de forma a ser reutilizável para todos os departamentos, utilizando uma estrutura de lista reversível genérica. Assim, a mesma estrutura de dados poderá ser aplicada para os diferentes tipos de produtos, permitindo flexibilidade e eficiência na gestão das informações.

2 Lista Reversível

Uma **Lista Reversível** é uma lista linear (não circular) e simplesmente encadeada, sem nós sentinelas. Ela contém elementos em ordenados de forma não decrescente de chave de busca e suporta as operações usuais de busca, inserção e exclusão, além de duas novas operações chamadas **undo** (desfazer) e **redo** (refazer). Vamos nos referir às operações de inserção e exclusão coletivamente como operações de edição. Uma lista é considerada a versão mais recente após aplicarmos uma operação de edição.

A operação de **undo** permite voltar a lista a uma versão anterior, revertendo a última operação de edição. Podemos chamar o **undo** n vezes para reverter as últimas n operações de edição. Se não houver uma operação de edição anterior (porque a lista é nova ou porque todas as operações de edição foram desfeitas), o **undo** não faz nada.

A operação de **redo** cancela a operação de **undo** anterior e é usada para atualizar a lista de volta para a versão mais recente. Podemos chamar o **redo** várias vezes consecutivas para cancelar várias operações de **undo**. O **redo** não tem efeito sobre a versão mais recente da lista.

Para suportar as operações de **undo** e **redo** em tempo constante, você precisa manter cópias das versões anteriores da lista. Assim, o **undo** e o **redo** podem simplesmente alternar entre diferentes versões da lista. Para reduzir os requisitos de memória, você deve manter apenas um número finito das versões mais recentes. Tentar desfazer além da versão mais antiga mantida não terá efeito sobre a lista.

Criar uma cópia de cada nó na lista após uma operação de edição não é necessário. Você deve compartilhar nós entre diferentes versões da lista sempre que possível. A figura a seguir ilustra isso com um exemplo de aplicação da lista para elementos do tipo inteiro. A Figura 1(a) mostra a lista original [2,5,6,7], apontada pelo ponteiro de cabeça h. A Figura 1(b) mostra a nova lista [2,6,7] após excluir(5), apontada pelo ponteiro de cabeça h'. Como os nós 6 e 7 permanecem inalterados, eles são compartilhados tanto pela lista [2,5,6,7] quanto pela [2,6,7].

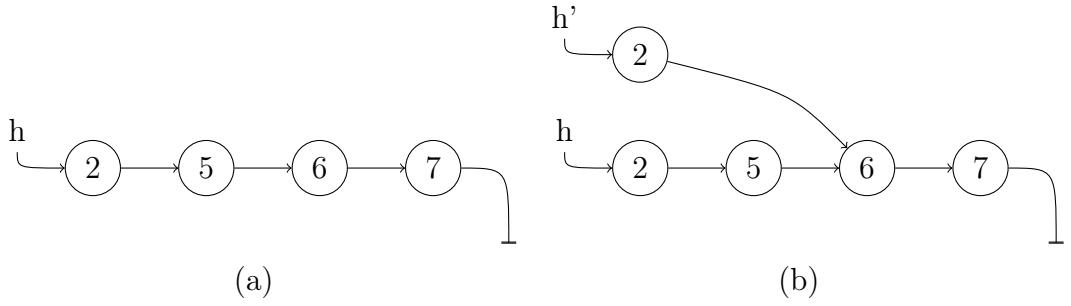

Figura 1: Exemplo de uma Lista Reversível.

3 Objetivo do Trabalho

Implemente o programa solicitado por Vianne. Para isso, implemente a Lista Reversível descrita na Seção 2 para **tipos genéricos de dados** e utilize a estrutura para implementação de quatro sistemas de gerenciamento (um sistema para o departamento de chocolates, um para o departamento de livros, um para o departamento de vinhos e um para o departamento de

streaming de filmes). O programa deve fornecer um menu principal para que o usuário escolha o departamento no qual ele deseja trabalhar. Cada sistema deve ter um menu com as opções: (1) Inserir; (2) Excluir; (3) Buscar; (4) Imprimir; (5) Undo; (6) Redo; (7) Sair. Quando o usuário escolher a opção para sair do sistema de um departamento, o programa deve voltar para o menu principal do programa. Neste, deve haver uma opção Sair que, quando selecionada, o programa termina. Para alternar do sistema de um departamento para outro, o usuário deverá escolher a opção Sair (do departamento corrente) e, no menu principal do programa, ele deve selecionar o departamento que ele deseja. Quando o usuário escolher a opção Sair do menu principal do programa, um relatório com todos os produtos de cada departamento deve ser impresso na tela.

4 Relatório e Entrega do trabalho

Além do **código fonte** e arquivo makefile, devem ser entregues os **slides** que serão apresentados em sala de aula e um **relatório** com a documentação do trabalho. Caso o programa não compile no Linux, a nota do trabalho será zero. No relatório deverão constar:

- A apresentação de todas as estruturas implementadas e a relação entre tais estruturas no trabalho.
- A descrição de cada função implementada (divididas em trechos de códigos ou apresentada integralmente) com a explicação de cada caso analisado.
- Exemplos de testes do programa onde todos os casos possíveis de entrada de dados são testados e o relatório é gerado.

Todas as implementações deverão ser desenvolvidas em grupo de **até três alunos** e deverá ser entregue até o dia 09/03/2025 às 23:59 hs no sistema AVA: **https://ava.ufes.br**.

Todos os arquivos devem ser compactados em um arquivo .zip. O nome do arquivo deve seguir o seguinte formato: GrupoX[nome-do-aluno-1][nome-do-aluno-2][nome-do-aluno-3].zip. Onde X deve ser substituído pelo número do grupo.

5 Avaliação do trabalho

A apresentação do trabalho será realizado nos dias 10 e 11/03/2025, nos horários e locais das aulas. A ordem de apresentação dos grupos será definida por sorteio. Todos os membros do grupo devem participar da apresentação. Caso algum membro não participe da apresentação, o grupo perderá pontos. Se o grupo não apresentar o trabalho a nota será zero.