INSTITUTO FEDERAL DE ALAGOAS – IFAL / CAMPUS MACEIÓ
Curso Bacharelado em Sistemas de Informação
Disciplina de Estruturas de Dados

Prof. MSc. Ricardo Nunes
Ricardo (arroba) ifal.edu.br

Miniprojeto: Sistema de Gerenciamento de Liderança Rotativa

Descrição: O projeto consiste em implementar um sistema de gerenciamento de ponto para reuniões de um grupo de estudo, no qual os membros se revezam de forma cíclica nas responsabilidades de liderança. A implementação usará uma estrutura de dados **lista encadeada circular** para manter a rotação dos membros de maneira eficiente.

Realização e Entrega

- A partir de *22/10 até 29/10*.
- Pode ser realizado *individualmente* ou em equipe com até *3 pessoas*.
- **O aluno deverá optar entre fazer o miniprojeto ou o miniteste sobre Listas Encadeadas.**
- Conteúdo a ser entregue:
  - Arquivos do código-fonte ou link (GitHub, Replit, etc.).
  - Arquivo **leiame.txt** contendo:
    - Nomes dos integrantes da equipe.
    - Lista de materiais consultados para realizar o projeto (sites, livros, artigos, etc.).
    - Comentário da equipe sobre o que foi alcançado no projeto.
    - Observações sobre problemas encontrados ou dificuldades na implementação, e funcionalidades que deveriam ser melhoradas.
  - Discussão sobre a estrutura da **lista encadeada circular** e seu desempenho no contexto do projeto.
- Apresentar o projeto funcionando ao professor.

Atividade 01: Implementar Lista Encadeada Circular

- Implementar uma **lista encadeada circular** para gerenciar os membros do grupo.
- Operações obrigatórias:
  - `adicionar_membro(membro)` : Adiciona um novo membro no final da lista.
  - `remover_membro(nome)` : Remove um membro pelo nome.
  - `proximo_responsavel()` : Exibe o próximo responsável pela reunião e atualiza a lista.

Estrutura Esperada:

1. **Classe** `Membro` : Representa cada membro do grupo de estudo, com atributos como nome e e-mail.
2. **Classe** `ListaEncadeadaCircular` : Implementa a lista circular com métodos para adicionar, remover membros e selecionar o próximo responsável.

Exemplo Funcionamento da Lista Circular

Exemplo demonstrando as operações básicas de **adicionar**, **remover** e **acessar o próximo** elemento em uma lista circular.

Operações

1. **Adicionar**: Inserimos novos elementos na lista.
2. **Mostrar Lista**: Exibimos todos os elementos presentes.
3. **Próximo**: Movemos para o próximo elemento da lista circular.
4. **Remover**: Removemos um elemento específico.

Exemplo

Aqui está o exemplo revisado com as operações solicitadas, utilizando uma **lista encadeada circular**:

Exemplo de Operações em uma Lista Encadeada Circular

1. **Adicionar Abel**:
   - Lista após a adição: `Abel`
   - Lista circular: `Abel -> (volta para Abel)`
2. **Próximo da lista**:
   - Elemento atual: `Abel` (único elemento, a lista retorna a Abel)
3. **Adicionar Bia**:
   - Lista após a adição: `Abel -> Bia`
   - Lista circular: `Abel -> Bia -> (volta para Abel)`
4. **Próximo da lista**:
   - Elemento atual: `Bia` (após Abel)
5. **Próximo da lista**:
   - Elemento atual: `Abel` (após Bia, voltamos ao início da lista)
6. **Adicionar Carlos**:
   - Lista após a adição: `Abel -> Bia -> Carlos`
   - Lista circular: `Abel -> Bia -> Carlos -> (volta para Abel)`
7. **Próximo da lista**:
   - Elemento atual: `Bia` (após Abel)
8. **Próximo da lista**:
   - Elemento atual: `Carlos` (após Bia)
9. **Adicionar Davi**:
   - Lista após a adição: `Abel -> Bia -> Carlos -> Davi`
   - Lista circular: `Abel -> Bia -> Carlos -> Davi -> (volta para Abel)`
10. **Remover Bruno**:
    - A operação falha porque "Bruno" não está na lista.
    - Lista permanece: `Abel -> Bia -> Carlos -> Davi -> (volta para Abel)`
11. **Próximo da lista**:
    - Elemento atual: `Abel` (após Davi, voltamos ao início da lista)

Resumo da Lista após todas as operações:

- A lista circular final é: `Abel -> Bia -> Carlos -> Davi -> (volta para Abel)`.
- Os elementos vão sendo acessados de forma cíclica conforme a função `proximo_da_lista()` é chamada, sempre retornando ao início da lista ao chegar no último elemento.

Atividade 02: Discussão sobre Desempenho

- Acrescentar no arquivo **leiame.txt** uma discussão sobre o desempenho de cada operação implementada no projeto.

Critérios de Avaliação

1. O código deve rodar sem erros.
2. Implementar corretamente a lista encadeada circular.
3. Seguir a especificação e atender aos requisitos do projeto.
4. Testar adequadamente as operações implementadas.
5. Incluir uma reflexão sobre o desempenho das operações.
6. Indicar materiais consultados no desenvolvimento do projeto.

Conjunto de Testes:

Teste de adição de membros:

- Adicionar diversos membros e verificar se o próximo responsável está seguindo o revesamento de forma cíclica.

Teste de remoção de membros:

- Remover um membro e garantir que ele não aparece mais na rotação.

Teste de próximo responsável:

- Garantir que após a execução da função `proximo_responsavel()`, o próximo membro na lista seja selecionado corretamente.