Sistema de Gerenciamento de Registros

Wilson Albuquerque Ramos
DRE:118092402
Link Repositório: https://github.com/wilsonufrj/ABBIndice

Estruturas de Dados Utilizadas

O projeto faz uso das seguintes estruturas:

- **std::vector**: Armazena os registros de forma contígua na memória, facilitando o acesso direto por índice e permitindo percursos eficientes.

- **Árvore Binária de Busca (ABB)**: Implementada na classe `ArvoreBinariaBusca`, indexa os registros por CPF, permitindo operações de inserção, busca e remoção com complexidade média de tempo O(logn)O(\log n)O(logn).

- **ABBIndice**: Especialização da ABB para armazenar pares chave-índice (CPF e posição no vetor), permitindo acesso rápido ao índice do registro no vetor principal.

Divisão de Módulos

O projeto foi organizado de forma modular, visando facilitar a manutenção e a legibilidade do código:

- **Registro**: Encapsula os dados de uma pessoa e o status de deleção.

- **No**: Estrutura básica de nó para a ABB.

- **ArvoreBinariaBusca**: Implementação genérica da ABB.

- **ABBIndice**: Extensão da ABB para indexação dos registros.

- **SistemaArquivosRegistros**: Coordena o vetor e o índice, implementando as operações principais do sistema.

- **main.cpp**: Responsável por realizar testes e demonstrar as funcionalidades.

Descrição das Rotinas e Funções

- **inserirRegistro**: Insere um novo registro, validando o CPF e garantindo unicidade.

- **buscarPorCpf**: Localiza um registro no vetor utilizando a ABB para acesso rápido.

- **removerPorCpf**: Marca o registro como deletado e remove a chave da ABB.

- **gerarEDLOrdenada / imprimirOrdenados**: Produzem e exibem uma lista ordenada dos registros ativos.

- **imprimirTodos**: Lista todos os registros, incluindo os marcados como deletados.

Complexidade de Tempo e Espaço

- **Tempo médio de busca, inserção e remoção**: O(logn)O(\log n)O(logn), assumindo árvore balanceada.

- **Pior caso**: O(n)O(n)O(n), se a árvore estiver desbalanceada (ex: inserções ordenadas).

- **Espaço**: O(n)O(n)O(n) para o vetor e O(n)O(n)O(n) para a ABBIndice.

Problemas e Observações

- Como a ABB não implementa balanceamento (não é AVL ou Red-Black), pode sofrer degradação de desempenho em casos patológicos de inserção ordenada.

- A validação do CPF considera apenas o tamanho, podendo ser melhorada para verificar dígitos verificadores.

- A marcação de registros como deletados mantém os dados no vetor, o que pode levar ao uso extra de memória ao longo do tempo.

Conclusão

O projeto atendeu ao objetivo de implementar um sistema simples e funcional de gerenciamento de registros com operações eficientes na média dos casos. A organização modular e o uso de templates facilitaram a reutilização e a especialização das classes. Futuras melhorias podem incluir o uso de uma árvore balanceada e otimizações no gerenciamento de memória.