Implementação de Tabela Hash em C

Aluno: Gabriel Baeta Pereira

1. Visão Geral do Projeto

Este projeto consiste na implementação de uma estrutura de dados de Tabela Hash em C, utilizando a técnica de endereçamento aberto, projetada para gerenciar de forma eficiente um grande volume de registros em memória secundária. A estrutura é composta pelas operações fundamentais de inserção, busca e remoção, que são a base para a manipulação dos dados na tabela.

A implementação foi modularizada em três arquivos principais (Hash.h, Hash.c, main.c).

2. Decisões de Design e Arquitetura

2.1. Estruturas de Dados e Arquivos

A fundação do projeto reside na struct Dados e na forma como o arquivo de hash é organizado:

- Dados: Modela o registro apresentado. A escolha por arrays de char de tamanho fixo (cpf[13], nome[50]) é uma abordagem direta para o escopo do trabalho, evitando a complexidade do gerenciamento dinâmico de memória para cada registro.
- **Arquivo de Hash (**hash.dat**)**: Atua como a própria tabela hash. É um arquivo binário que armazena um grande array de Dados. A posição de cada registro no arquivo corresponde a um slot na tabela.
- **Marcação de Slots Vazios**: Uso de um valor especial (cpf = "-1") para indicar slots vazios ou removidos. Esta marcação é fundamental para o funcionamento correto dos algoritmos de busca e inserção em um esquema de endereçamento aberto.

2.2. Gerenciamento de Arquivos

Foi adotada uma separação entre o arquivo de dados inicial e o arquivo que representa a estrutura de índice (a tabela hash):

- dados.dat: Contém a massa de dados original com os registros completos dos alunos.
- hash.dat: Armazena a Tabela Hash. É o arquivo de índice onde os registros são organizados de acordo com a função de hash para permitir buscas rápidas.

3. Análise dos Algoritmos Implementados

A implementação utiliza **Endereçamento Aberto** com uma sondagem pseudo-aleatória para resolver colisões.

3.1. Inserção (insere)

A inserção aplica a função de hash para encontrar uma posição para o novo registro,a função h() calcula uma posição inicial, se o slot estiver ocupado, o algoritmo testa novas posições em uma sequência de sondagem, e o algoritmo continua a sondagem até encontrar um slot vazio, onde o novo registro é finalmente inserido. Antes da inserção, é verificado se um registro com o mesmo CPF já existe para evitar duplicatas.

3.2. Busca (busca)

A busca emula o processo de inserção para localizar um registro, partindo da posição inicial calculada pela função h(), o algoritmo percorre a mesma sequência de sondagem da inserção.

3.3. Remoção (remover)

A remoção é a operação mais delicada no endereçamento aberto e implementa os seguintes passos:

1. **Localização**: A chave é primeiro localizada usando o algoritmo de busca padrão.
2. **Remoção Lógica**: Em vez de apagar o registro, o que quebraria a sequência de sondagem para outros registros, o slot é sobrescrito com um marcador especial.
3. **Manutenção da Integridade**: Essa abordagem garante que futuras buscas por chaves que colidiram com o elemento removido possam "pular" o slot agora vazio e continuar a sondagem corretamente.

3.4. Testes (main.c)

A função main foi projetada não apenas como um ponto de entrada, mas como um **ambiente de testes robusto**. A metodologia de primeiro gerar um arquivo com 10.000 registros, construir a tabela hash a partir dele. Além disso, o sistema permite que o usuário teste as funcionalidades de inserir, buscar e remover alunos por CPF em tempo real

https://github.com/GabrielBaetaP/Trab_EDA/tree/main/TRAB_EDA_HASH