# Tabela Hash

Arthur Barreto Godoi e Ricardo Quer Do Nascimento Filho
*Departamento de Informática*
*Universidade Federal do Paraná – UFPR*
Curitiba, Brasil

***Resumo*—Implementação de uma tabela hash de endereçamento aberto juntamente com sua inclusão e exclusão.**

## I. Introdução

Este projeto consiste na implementação de uma estrutura de dados chamada **Tabela Hash em linguagem C**. A tabela hash é uma estrutura de dados amplamente utilizada para armazenar pares chave-valor, oferecendo uma forma eficiente de inserir, buscar e remover dados com complexidade média constante, O(1), para cada operação. Neste código, a tabela hash foi construída usando o método de hash duplo, que emprega duas funções de hash diferentes para distribuir os elementos em duas tabelas separadas (T1 e T2). Esse método serve para reduzir o número de colisões, ou seja, situações em que duas chaves diferentes são mapeadas para a mesma posição na tabela.

## II. Estrutura do código

O código está estruturado em três arquivos principais: **hash.h, hash.c, e main.c**. O arquivo de cabeçalho (hash.h) contém as definições e assinaturas de funções, enquanto hash.c contém a lógica da implementação das funções, e main.c é responsável pela interação com o usuário e execução das operações na tabela hash.

1) **Estrutura da Tabela Hash:** A estrutura HashTable define duas tabelas (T1 e T2), ambas de tamanho fixo M = 11. Cada posição das tabelas é inicializada com EMPTY, indicando que está vazia, ou com DELETED, sinalizando que um elemento foi removido.

2) **Funções de Hash:**
   a) **h1:** A função h1 é uma função de hash simples que calcula a posição usando o módulo (k mod M), onde k é a chave e M o tamanho da tabela.

   b) **h2:** A função h2 aplica uma fórmula alternativa para distribuir chaves. Ela calcula a posição baseada na multiplicação da chave por um fator decimal, ajudando a distribuir os elementos de forma mais uniforme em T2.

3) **Inicialização:** A função initialize table percorre cada posição das tabelas T1 e T2, atribuindo o valor EMPTY, para podermos identificar posições vazias.

4) **Inserção:** A função insert tenta colocar a chave k diretamente em T1, na posição indicada por h1. Se essa posição estiver ocupada (colisão), o valor já presente em T1 é deslocado para T2 na posição indicada por h2, e k é inserido em T1. Esse mecanismo permite gerenciar colisões sem precisar redimensionar as tabelas.

5) **Remoção:** A função remove key usa h1 e h2 para verificar se a chave está presente em T1 ou T2. Caso a chave esteja na posição esperada, ela é marcada como DELETED ou EMPTY, respectivamente.

6) **Busca:** A função find permite localizar uma chave em uma das tabelas, retornando a posição da chave se ela for encontrada. Essa função usa o índice i para decidir qual função de hash usar (0 para h1 e 1 para h2), garantindo flexibilidade na busca.

7) **Impressão Ordenada:** Para facilitar a visualização dos elementos armazenados, as funções print hash1 e print hash2 imprimem os elementos de T1 e T2, respectivamente, em ordem crescente. Para isso, essas funções implementam o algoritmo de ordenação por seleção (Selection Sort). Após ordenar os elementos, cada valor é impresso, indicando a tabela e a posição calculada pela respectiva função de hash.

8) **Função Principal (Main):** Em main.c, o programa inicializa a tabela e aguarda comandos do usuário via entrada padrão (stdin). Dependendo da operação (i para inserir e r para remover), o programa executa as funções correspondentes. No final, as duas tabelas são exibidas, mostrando o conteúdo de T1 e T2.

## III. Conclusão

Esta implementação de tabela hash em C utiliza duas funções de hash e duas tabelas para lidar com colisões, melhorando a distribuição das chaves e diminuindo os conflitos. O uso de hash duplo torna o gerenciamento de colisões mais eficiente, enquanto a estrutura de dados permanece simples e eficiente em termos de memória. Além disso, a inclusão de funções para ordenação e impressão permite uma visualização clara dos elementos da tabela, facilitando a depuração e o

entendimento da estrutura de dados. Este projeto exemplifica uma aplicação básica de hash em C, abordando aspectos fundamentais de desempenho e organização de dados em estruturas de tamanho fixo.