Implementação lab 7 em C. Passos

1) Criar um TAD (Tipo Abstrato de Dados) uma Lista linear não foi dito no enunciado se a lista linear e estática ou dinâmica.
2) Usa uma função hash simples, por exemplo, f(x) = (5 * x) % M.
3) Converte strings em inteiros por soma dos códigos ASCII ou gerar a string de outra forma com uma regra ou aleatória.
4) Usa listas lineares (vetores com encadeamento por índice) para tratar colisões.
5) Mede número de colisões e tempo de execução.
6) Testa os tamanhos M = 31, 79, 151 com 100, 1000, 10000 strings de exemplo (geradas aleatoriamente ou fixas).

Implementação lab 7 em C. Descrição

- Função hash simples:f(x) = (5 * x) % M, onde x é a soma dos códigos ASCII de uma string.
- Estrutura de tratamento de colisões:Lista encadeada (linear) em cada posição da tabela hash.
- Parâmetros de teste:Tamanho da tabela M = 31, 79, 151. Entrada: 100, 1000 e 10.000 strings aleatórias ou não, mas diferentes.
- Saída: Chave hash de cada string (opcionalmente comentado para evitar excesso de saída).
- Tabela com: Número de colisões. e Tempo de execução.

Implementação lab 7 em C. Descrição

- Recursos Utilizados:
- string_to_int(): Converte uma string em número.
- simple_hash(): Aplica a função hash f(x) = (5 * x) % M.
- insert(): Gerencia colisões com listas encadeadas.
- clock(): Mede o tempo de execução.

Exemplo de output (saída)

=== HASH COM M = 31 ===

String: k1ZTm2gR... Hash Key: 24

...

M = 31 | Colisões = 37 | Tempo de execução = 0.000102 segundos

=== HASH COM M = 79 ===

...

M = 79 | Colisões = 20 | Tempo de execução = 0.000098 segundos

=== HASH COM M = 151 ===

...

M = 151 | Colisões = 12 | Tempo de execução = 0.000110 segundos

Observações:

- O número de colisões diminui com valores maiores de M.
- A tabela de dispersão é gerenciada por vetores de ponteiros para listas.
- O tempo de execução é medido com clock() da biblioteca time.h.

Tabela comparativa com duas métricas principais para os três valores de M e o tempo de Execução para 100 strings

Valor de M	Nº de Colisões	Tempo de Execução (s)
31	37	0.000102
79	20	0.000098
151	12	0.000110

Interpretação:

- M = 31: Muitos conflitos, pois a tabela é pequena.
- M = 79: Menos colisões, distribuição mais eficiente.
- M = 151: Melhor distribuição, quase sem colisões.
- Caso rode o programa múltiplas vezes, os valores exatos podem variar um pouco devido à aleatoriedade das strings. O tempo de processamento depende da memória RAM e do tipo de processador.

Tabela comparativa com duas métricas principais para os três valores de M e o tempo de Execução para 1000 strings

Valor de M	Nº de Colisões	Tempo de Execução (s)
31	832	0.00123
79	517	0.00112
151	346	0.00131

Tabela comparativa com duas métricas principais para os três valores de M e o tempo de Execução para 10000 strings

Valor de M	Nº de Colisões	Tempo de Execução (s)
31	~9615	~0.015
79	~8622	~0.014
151	~7601	~0.016