Tabela Hash

Eduardo Stefanel Paludo (GRR20210581)
Fernando Gbur dos Santos (GRR20211761)
Universidade Federal do Paraná (UFPR)
Curitiba, Brasil

I. Introdução

O projeto desenvolvido consiste na implementação de uma tabela Hash de endereçamento aberto na linguagem C. Para isso, foram escritos seis arquivos: o programa principal (main.c), uma biblioteca com as principais funções da hash (hash.c), uma biblioteca com funções de ordenação (sort.c), dois arquivos de header (hash.h e sort.h), e um Makefile. O programa foi compilado através do comando *make*, com os parâmetros *-Wall* e *-std=c99*.

As funções implementadas foram: h1 (cálculo para o índice da tabela 1), h2 (cálculo para o índice da tabela 2), hash_init, item_init, hash_search, hash_insert, hash_remove, hash_delete, hash_print, hash_merge, sort, insert, search e shift. Após a conclusão do projeto, foram comparadas as saídas para cada caso de teste com a saída esperada do programa.

II. Programa principal

No arquivo principal (*main.c*), são inicializadas as 2 tabelas utilizadas na implementação da *Hash* e o vetor de itens auxiliar para realizar a ordenação e impressão. Em seguida, para ler o arquivo recebido pela linha de comando, foi utilizado um *while (!feof(stdin))*, em conjunto com um *scanf* que lê um caractere. Quando o caractere lido corresponde à instrução de inserir na hash(i), um inteiro é lido e inserido nela. O mesmo ocorre com a instrução de remoção.

Após a leitura do arquivo, todos os itens existentes nas duas tabelas são inseridos no vetor auxiliar e ordenados por meio do algoritmo de ordenação *Insertion Sort* (pelo valor). Em seguida, o valor de cada item, a tabela onde ele foi inserido e o índice da mesma são impressos no terminal. Por fim, todas as estruturas são excluídas e o programa encerra.

III. Estrutura

Para implementar a tabela Hash, foram utilizados dois vetores para as tabelas e uma *struct* correspondente a um item, contendo o índice onde o item foi armazenado na tabela (*int*), uma variável que diz se o valor foi removido ou não (*int*), o valor armazenado (*int*) e em qual tabela ele está armazenado (*char**). A função hash_init aloca memória para uma nova tabela vazia e aguarda as inserções.

IV. Inserção

A inserção na Hash ocorre da seguinte forma: primeiro, é calculado o índice onde o valor vai ser inserido na T1, através da função *k mod m* (k = valor a ser inserido, m = 11). Se não existir nenhum item no índice obtido, um novo item é inicializado com as suas respectivas propriedades e inserido na tabela. Por outro lado, se já existir algum item neste índice da T1, é calculado outro índice para este item já existente para ser inserido na T2, por meio da função *floor(m * (k * 0.9 - floor(k * 0.9)))*. Se não tiver nenhum item alocado nesta posição da T2, ele é criado, armazenado nela e o valor originalmente inserido ocupa seu antigo lugar na T1.

V. Remoção

A remoção ocorre de forma mais simples comparada à inserção. Ela recebe o valor a ser removido e passa pelas duas funções das tabelas T1 e T2. Com os índices calculados, o algoritmo primeiramente verifica na tabela T2. Se no índice calculado existir um item e o valor desse item for igual ao valor passado para a função, o item é desalocado e sua posição na tabela é igualada a *NULL*.

Se o valor não existir em T2, ele é procurado em T1. Se existir item em T1 e seu valor for igual ao passado para a função, sua variável de removido é igualada a 1, mas o item não é desalocado. Isso acontecerá apenas no final do programa, facilitando próximas inserções na mesma posição.

VI. Ordenação e Impressão

Para a impressão ordenada dos elementos restantes da tabela, foi criado um vetor de itens auxiliar. Após todas as operações serem realizadas, o algoritmo percorre as duas tabelas, transfere seus itens alocados e não removidos para este vetor auxiliar e ordena o vetor utilizando o algoritmo de ordenação *Insertion Sort*, utilizando como parâmetro de comparação o valor armazenado no item.

Ocorre a impressão das propriedades do item na seguinte ordem: valor, tabela e endereço. Todas as estruturas são desalocadas e o programa se encerra.

VII. Conclusão

Após a conclusão dos algoritmos, foram executados testes conforme especificado, utilizando os comandos *./myht* e *diff*. As saídas obtidas foram compatíveis com as saídas esperadas. Além disso, não foi observado nenhum erro de alocação de memória e inicialização de variáveis, através da execução do programa pelo *Valgrind*.