Tabela Hash

João Armênio Silveira
Departamento de Informática
Universidade Federal do Paraná – UFPR

***Resumo*—Este relatório descreve a implementação de duas tabelas hash de tamanho m = 11, com o objetivo de armazenar e recuperar valores de forma eficiente.**

I. Introdução

A tabela Hash constitui na implementação de um dicionário que responde a uma chave dada, e após a aplicação da função hash selecionada, é armazenada no index da tabela.

Nessa implementação em específico são codificadas duas tabelas (T1 e T2), ambas de tamanho m = 11. Dentro de cada espaço disponível na tabela é alocada uma estrutura que indica, a "data"(chave enviada), "table"(tabela, T1 ou T2) e "index"(index da tabela que a contém), como mostra a imagem. Junto a isso foram implementadas as funções de criação e destruição das tabelas, inserção e remoção e de impressão final do resultado pós-operações.

```
typedef struct HtItem {
    int table;
    int data;
    int index;
} HtItem;
```

Figura 1. Struct item da tabela hash.

II. Funções Implementadas

Neste relatório, foi implementado de forma convencional todas as funções necessárias para o funcionamento das duas tabelas hash. Isso inclui inserção de valores, busca por valores e tratamento de colisões.

Quando ocorrem colisões na primeira tabela hash, foi implementada a técnica de passar o valor da chave para a segunda tabela, e então atualizar os atributos da estrutura.

Nesse as soluções para colisões foram implementadas de forma clássica e sem nenhum tipo de personalização ou abordagem alternativa.

```
HtItem *createTable();
void freeTables(HtItem *T1, HtItem *T2);
int insertHt(HtItem *T1, HtItem *T2, int key);
int removeHt(HtItem *T1, HtItem *T2, int key);
void printTables(HtItem *T1, HtItem *T2);
```

Figura 2. Funções implementadas.

A maior particularidade da implementação é feita no momento de impressão das tabelas. Com o objetivo de realizar a impressão em ordem crescente com base nas chaves foi criado um array da estrutura HtItem, onde foram adicionados todos elementos que não eram vazios ou excluídos das outras duas tabelas. Então essa array foi ordenada usando a função qsort nativa do C. E enfim o resultado sai conforme especificado.

III. Particularidades

O principal destaque da implementação está relacionado a impressão das tabelas. Para imprimir as chaves em ordem crescente de acordo com as chaves, foi criado um array da estrutura HtItem. Nesse array, foram adicionados todos elementos presentes nas tabelas, desconsiderando os espaçoes vazios e os elementos excluídos. Em seguida, o array foi ordenado utilizando a função nativa qsort do C, resultando na impressão conforme o especificado.

IV. A Função Main

A execução é simples, o arquivo principal (main.c) recebe as entradas, chama as funções insertHt ou removeHt de acordo com a leitura do teste enviada.

```
int main()
{
    HtItem *T1 = createTable();
    HtItem *T2 = createTable();

    char command;
    int value;

    fscanf(stdin, "%c %d", &command, &value);
    getc(stdin);

    while (! feof(stdin)){
        if (command == 'i')
            insertHt(T1, T2, value);
        else if (command == 'r')
            removeHt(T1, T2, value);

        fscanf(stdin, "%c %d", &command, &value);
        getc(stdin);
    }

    printTables(T1, T2);

    freeTables(T1, T2);
}
```

Figura 3. Função main.