Tabela Hash Com Leitura De Arquivo

O código implementa uma tabela hash para armazenar informações de alunos, incluindo matrícula e nome, usando encadeamento separado para lidar com colisões. Nesse método, cada posição na tabela hash mantém uma lista encadeada de elementos que mapeiam para aquela posição.

O código

main.c: O arquivo principal que contém a função main(). Ele configura a localização e o tamanho da tabela hash, inicializa a tabela, exibe um menu interativo para interação com o usuário e gerencia as ações escolhidas no menu.

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include "hash.h"

int main() {
    setlocale(LC_ALL, "Portuguese");
    int tam = tamanhoVetor();
    TDados *hash[tam];
    inicializa(hash, tam);
    menu(hash, tam);
    return 0;
}
```

hash.c: Contém a implementação das funções relacionadas à manipulação da tabela hash e dos elementos nela contidos.

inicializarTabela(): Inicializa a tabela hash, definindo todas as posições como vazias (NULL).

```
void inicializarTabela(TDados **hash, int tam) {
        int i;
    for (i = 0; i < tam; i++) {
        hash[i] = NULL;
    }
}
```

calcularTam(): Calcula o tamanho do vetor hash com base no número de elementos no arquivo de entrada.

```
int calcularTam(int tam) {
   float metade = (float)(tam / 2) * 1.5;
   return metade;
}
```

calcularTamanhoTabela(): Calcula o tamanho do vetor hash baseado nos dados do arquivo de entrada.

```
int calcularTamanhoTabela() {
   int i = 0, size = 0;
   char vetor[100], parag = '\n';
   FILE *aluno;
   aluno = fopen("lista_matricula_alunos_2023.txt", "r");
   if (!aluno)
      printf("\nErro na abertura do arquivo 'lista_matricula_alunos_2023.txt'");
   else {
      while (fgets(vetor, sizeof(vetor), aluno)) {
         if (strlen(vetor) > 1) {
            size++;
         }
      }
      size++;
      fclose(aluno);
   }
   size = calcularTam(size);
   return size;
}
```

inserirDadoLista(): Insere um novo elemento (aluno) na lista encadeada da posição correspondente na tabela hash.

```
void inserirDadoLista(TDados **hash, long long matricula, string nomeAluno, int tam) {
    int pos = calcularMod(matricula, tam);
    TDados **lista = &hash[pos];

    if (*lista == NULL) {
        *lista = (TDados *)malloc(sizeof(TDados));
        if (*lista == NULL) {
            printf("\nErro na alocacao da memoria!");
            return;
        }
        strcpy((*lista)->nome, nomeAluno);
        (*lista)->matricula = matricula;
        (*lista)->prox = NULL;
    } else {
        TDados *armazena = hash[pos];
        while ((*lista)->prox != NULL)
            *lista = (*lista)->prox;
        (*lista)->prox = (TDados *)malloc(sizeof(TDados));
        if ((*lista)->prox == NULL) {
            printf("\nErro na alocacao da memoria!");
            return;
        }
        *lista = (*lista)->prox;
        strcpy((*lista)->nome, nomeAluno);
        (*lista)->matricula = matricula;
        (*lista)->prox = NULL;
        hash[pos] = armazena;
    }
}
```

insereElemento(): Insere um novo elemento (aluno) na tabela hash, verificando se a matrícula já existe.

```
void insereElemento(TDados **hash, int tam, long long matricula, string nomeAluno) {
    int posicao = calcularMod(matricula, tam);
    if (hash[posicao] != NULL) {
        if (buscaLista(hash, matricula, tam)) {
            printf("\nMatricula ja cadastrada: %lld - %s\n", matricula, nomeAluno);
            return;
        }
    }
    inserirDadoLista(hash, matricula, nomeAluno, tam);
}
```

adicionarNovoAluno(): Permite inserir um novo elemento (aluno) a partir da entrada do teclado.

```
void adicionarNovoAluno(TDados **hash, int tam) {
    string nomeAluno;
    long long matricula;
    getchar();
    printf("\nInsira o nome: ");
    fgets(nomeAluno, sizeof(nomeAluno), stdin);
    printf("\nInsira a matricula: ");
    scanf("%lld", &matricula);
    insereElemento(hash, tam, matricula, nomeAluno);
}
```

leituraArq(): Lê os dados do arquivo de entrada e insere os elementos na tabela hash.

```
void leituraArq(TDados **hash, int tam) {
    long long matricula;
    string nomeAluno;
    FILE *aluno;

    printf("Abrindo arquivo para leitura...\n");

    aluno = fopen("lista_matricula_alunos_2023.txt", "r");
    if (!aluno) {
        printf("\nErro na abertura do arquivo 'lista_matricula_alunos_2023.txt'");
    } else {
        printf("Lendo dados do arquivo...\n");

        while (!feof(aluno)) {
            if (fscanf(aluno, "%lld", &matricula)) {
                fgetc(aluno); // Consumir o caractere '\n' após a leitura da matrícula
                fgets(nomeAluno, sizeof(nomeAluno), aluno);
                if (nomeAluno[strlen(nomeAluno) - 1] == '\n') {
                    nomeAluno[strlen(nomeAluno) - 1] = '\0'; // Remover o caractere '\n' do final
                }
                insereElemento(hash, tam, matricula, nomeAluno);
            }
        }
        fclose(aluno);
        printf("\n\nLeitura concluída. Dados inseridos na tabela hash.\n");
    }
}
```

calcularMod(): Calcula o índice onde um elemento deve ser inserido ou buscado.

```
int calcularMod(long long matricula, int tam) {
    int mod = matricula % tam;
    return mod;
}
```

buscaLista(): Verifica se um elemento (aluno) com uma determinada matrícula está na lista encadeada da posição correspondente na tabela hash.

```
int buscaLista(TDados **hash, long long matricula, int tam) {
    int posi = calcularMod(matricula, tam);
    TDados *lista = hash[posi];
    while (lista != NULL) {
        if (matricula == lista->matricula)
            return 1;
        lista = lista->prox;
    }
    return 0;
}
```

busca(): Busca e exibe um elemento (aluno) com uma determinada matrícula na tabela hash.

```
void busca(TDados **hash, long long matricula, int tam) {
    int posi = calcularMod(matricula, tam);
    int exibe = 0;
    TDados *lista = hash[posi];
    while (lista != NULL) {
        if (matricula == lista->matricula) {
            printf("\n - Hash[%d]: %lld - %s \n", posi,lista->matricula, lista->nome);
            exibe = 1;
        }
        lista = lista->prox;
    }
    if (!exibe)
        printf("\n\tMatricula nao existe");
}
```

percorreLista(): Percorre uma lista encadeada e escreve os elementos em um arquivo.

```
void percorreLista(TDados **hash, int pos, FILE *aluno){
    TDados *lista = hash[pos];
    while (lista != NULL) {
        fprintf(aluno, "%lld\n%s\n", lista->matricula, lista->nome); // Adicionar uma quebra de linha após
o nome
        lista = lista->prox;
    }
    fprintf(aluno, "\n"); // Adicionar uma linha em branco entre os registros
}
```

salvarArqHash(): Salva os elementos da tabela hash em um arquivo.

```
void salvarArqHash(TDados **hash, int tam) {
    FILE *aluno;
    int i;
    aluno = fopen("lista_matricula_alunos_2023.txt", "w");
    if (!aluno) {
        printf("\nErro na abertura do arquivo 'lista_matricula_alunos_2023.txt'");
    } else {
        for (i = 0; i < tam; i++) {
            TDados *lista = hash[i];
            while (lista != NULL) {
                fprintf(aluno, "%lld\n%s\n", lista->matricula, lista->nome);
                lista = lista->prox;
            }
        }
        fclose(aluno);
        printf("\nBase de dados salva com sucesso!\n");
    }
}
```

obterMatricula(): Obtém a matrícula a ser buscada ou inserida.

```
long long obterMatricula() {
    long long matricula;
    printf("Informe a matricula: ");
    scanf("%lld", &matricula);
    return matricula;
}
```

excluirElemento(): Remove um elemento (aluno) da tabela hash.

```
void excluirElemento(TDados **hash, long long matricula, int tam) {
    int pos = calcularMod(matricula, tam);
    if (!buscaLista(hash, matricula, tam)) {
        printf("\nItem nao encontrado\n");
        return;
    }
    TDados **lista = &hash[pos];
    TDados *anterior = *lista;
    TDados *proximo = (*lista)->prox;
    TDados *guarda = hash[pos];
    while (*lista != NULL) {
        if ((*lista)->matricula == matricula) {
            if (*lista == anterior) {
                if ((*lista)->prox == NULL) {
                    free(*lista);
                    hash[pos] = NULL;
                } else {
                    (*lista)->prox = proximo->prox;
                    strcpy((*lista)->nome, proximo->nome);
                    (*lista)->matricula = proximo->matricula;
                    hash[pos] = guarda;
                }
            } else {
                anterior->prox = proximo;
                free(*lista);
                hash[pos] = guarda;
            }
            return;
        }
        anterior = *lista;
        *lista = (*lista)->prox;
        proximo = (*lista)->prox;
    }
    hash[pos] = guarda;
    free(anterior);
    free(proximo);
    free(guarda);
}
```

imprimirElementosLista(): Imprime os elementos de uma lista encadeada.

```
void imprimirElementosLista(TDados *lista) {
    if (lista == NULL) {
        printf("");
    } else {
        while (lista != NULL) {
            printf("%lld - %s", lista->matricula, lista->nome);
            if (lista->prox != NULL) {
                printf("\n\t");
            }
            lista = lista->prox;
        }
    }
}
```

exibirLista(): Exibe todos os elementos da tabela hash.

```
void exibirLista(TDados **hash, int tam) {
        int i;
   for (i = 0; i < tam; i++) {
      printf("Hash[%d]: ", i);
      imprimirElementosLista(hash[i]);
      printf("\n");
   }
}
```

liberarMemoria(): Libera a memória alocada para os elementos da tabela hash.

```
void liberarMemoria(TDados **hash, int tam) {
        int i;

        printf("\nSaindo...\n");

   for (i = 0; i < tam; i++) {
      TDados *lista = hash[i];
      while (lista != NULL) {
         TDados *temp = lista;
         lista = lista->prox;
         free(temp);
      }
   }
}
```

executarAcaoMenu(): responsável por executar as ações correspondentes à opção escolhida pelo usuário no menu.

```
void executarAcaoMenu(TDados **hash, int num, int tam) {
    long long info;
    switch (num) {
        case 0:
          liberarMemoria(hash, tam);
           break;
        case 1:
           adicionarNovoAluno(hash, tam);
           break;
        case 2:
           info = obterMatricula();
           excluirElemento(hash, info, tam);
           break;
        case 3:
           info = obterMatricula();
           busca(hash, info, tam);
           break;
        case 4:
           exibirLista(hash, tam);
           break;
        case 5:
           salvarArqHash(hash, tam);
           break;
        default:
           printf("Nao existe essa opcao!");
           break;
    }
}
```

exibirMenu(): exibir um menu interativo para o usuário, permitindo que ele escolha diferentes ações para executar na tabela hash.

```
void exibirMenu(TDados **hash, int tam) {
    int op;
    leituraArq(hash, tam);
    do {
        printf("\n\n---------- MENU ----------\n\n");
        printf("\t0. Sair\n");
        printf("\t1. Inserir novo aluno\n");
        printf("\t2. Remover aluno\n");
        printf("\t3. Buscar aluno\n");
        printf("\t4. Imprimir a base de dados\n");
        printf("\t5. Salvar base de dados\n");
        printf("Escolha a opcao: ");
        scanf("%d", &op);
        executarAcaoMenu(hash, op, tam);
    } while (op != 0);
}
```

hash.h: Contém as definições das estruturas de dados e as assinaturas das funções utilizadas no código.

```
#ifndef ARQUIVOHASH_H
#define ARQUIVOHASH_H

typedef char string[100];

typedef struct tipoDados {
   long long matricula;
   string nome;
   struct tipoDados *prox;
} TDados;
```

```
void inicializarTabela(TDados **hash, int tam);
int calcularTam(int tam);
int calcularTamanhoTabela();
void inserirDadoLista(TDados **hash, long long matricula, string nomeAluno, int tam);
void insereElemento(TDados **hash, int tam, long long matricula, string nomeAluno);
void adicionarNovoAluno(TDados **hash, int tam);
void leituraArq(TDados **hash, int tam);
int calcularMod(long long matricula, int tam);
int buscaLista(TDados **hash, long long matricula, int tam);
void busca(TDados **hash, long long matricula, int tam);
void percorreLista(TDados **hash, int pos, FILE *aluno);
void salvarArqHash(TDados **hash, int tam);
long long obterMatricula();
void excluirElemento(TDados **hash, long long matricula, int tam);
void imprimirElementosLista(TDados *lista);
void exibirLista(TDados **hash, int tam);
void liberarMemoria(TDados **hash, int tam);
void executarAcaoMenu(TDados **hash, int num, int tam);
void exibirMenu(TDados **hash, int tam);

#endif
```

A estrutura TDados para representar um elemento (aluno) na tabela hash, contendo informações como matrícula, nome e um ponteiro para o próximo elemento na lista encadeada. A implementação permite inserir, buscar, excluir e listar elementos na tabela hash, além de salvar e carregar esses elementos de um arquivo.