



TRABALHO 03 - HASHING

Atenção

- **Prazo de entrega: 12/11/2015 – 23h55 (via Moodle).**
Após o prazo, o trabalho não será considerado!

Indexação usando Tabelas Hash

O seu sistema de cadastro de partidas profissionais de *League of Legends* está sendo utilizado em larga escala e agora você contratou uma equipe para dar continuidade e manutenção.

O analista de dados da sua equipe identificou que agora a maior parte das operações é de busca e que são realizadas poucas inserções ou remoções de registros. Sendo assim, concluiu-se que utilizar uma estrutura de *hashing* poderá trazer grandes benefícios ao desempenho do sistema, permitindo que a maioria das buscas seja realizada com poucos acessos ao disco.

Lembrando, cada partida (registro no arquivo de dados) é composta pelos seguintes campos:

- *Código* (composição de letras maiúsculas da primeira letra do nome da equipe azul, seguida da primeira letra do nome da equipe vermelha, seguida das duas primeiras letras do apelido do MVP e do dia e mês da partida, ex: **FTFE0705** - esse campo é a chave primária, portanto, não poderá existir outro valor idêntico);
- *Nome da equipe azul* (nome da equipe jogando no lado azul do mapa, ex: **Fnatic**);
- *Nome da equipe vermelha* (nome da equipe jogando no lado vermelho do mapa, ex: **Team Solo Mid**);
- *Data da partida* (data no formato DD/MM/AAAA, ex: 07/05/2015);
- *Duração da partida* (duração do jogo no formato MM:SS, ex: 31:47);
- *Nome da equipe vencedora* (nome da equipe que venceu a partida. Deve ser igual ao nome da equipe azul ou da equipe vermelha, ex: **Fnatic**);
- *Placar do jogo* (contagem de abates de cada time, sendo dois campos distintos com 2 dígitos inteiros cada ex: 15, 06);
- *apelido do MVP* (*Most Valuable Player* – apelido do jogador que mais se destacou na partida, ex: **Febiven**);

Garantidamente, nenhum campo de texto receberá caractere acentuado.

Tarefa

Desenvolva um programa que permita ao usuário manter uma base de dados de partidas. O programa deverá permitir:

1. Inserir uma nova partida;
2. Modificar o campo **duração** de uma partida a partir da chave primária;
3. Buscar partidas a partir de sua chave primária;
4. Remover partidas a partir de sua chave primária;
5. Listar a Tabela Hash.

Mais uma vez, nenhum arquivo ficará salvo em disco. O arquivo de dados será simulado em uma *string* e o índice primário será sempre criado na inicialização do programa e manipulado em memória RAM até o término da execução. Suponha que há espaço suficiente em memória RAM para todas as operações.

Arquivo de dados

Como este trabalho será corrigido pelo **OnlineJudge** e o sistema não aceita funções que manipulam arquivos, os registros serão armazenados e manipulados em uma *string* que simula o arquivo aberto. Você deve utilizar a variável global **ARQUIVO** e funções de leitura e escrita em strings, como **sprintf** e **sscanf**, para simular as operações de leitura e escrita em arquivo.

Dicas

- Você nunca deve perder a referência do começo do arquivo, então não é recomendável percorrer a *string* diretamente pelo ponteiro **ARQUIVO**. Um comando equivalente a **fseek(f, 192, SEEK_SET)** é **char *p = ARQUIVO + 192**.
- Diferentemente do **fscanf**, o **sscanf** não movimenta automaticamente o ponteiro após a leitura.
- O **sprintf** adiciona automaticamente o caractere **\0** no final da *string* escrita. Em alguns casos você precisará sobrescrever a posição.

O arquivo de dados deve ser organizado em registros de tamanho fixo de 192 bytes. Os campos *nome da equipe azul*, *nome da equipe vermelha*, *nome da equipe vencedora* e *apelido do MVP* devem ser de tamanho variável. Os demais campos devem ser de tamanho fixo: *data de nascimento* (10 bytes), *duração da partida* (5 bytes), *placar da equipe azul* (2 bytes), *placar da equipe vermelha* (2 bytes) e *chave primária* (8 bytes). A soma de bytes dos campos fornecidos (incluindo os delimitadores necessários) nunca poderá ultrapassar 192 bytes. Os campos do registro devem ser separados pelo caractere delimitador **@** (arroba). Cada registro terá 9 delimitadores, mais 27 bytes ocupados pelos campos de tamanho fixo. Você precisará garantir que os demais campos juntos ocupem um máximo

de 156 bytes. Caso o registro tenha menos de 192 bytes, o espaço adicional deve ser marcado com o caractere # de forma a completar os 192 bytes. Para evitar que o registro exceda 192 bytes, cada campo de tamanho variável deve ocupar no máximo 39 bytes. O programa deve impedir a inserção de registros com campos de mais de 39 bytes.

```
FTFE0705@Fnatic@Team SoloMid@07/05/2015@31:47@Fnatic@15@06@Febiv
en@#####
#####
SEDE1005@SKTelecom T1@Edward Gaming@10/05/2015@37:38@Edward Gami
ng@09@25@Deft@#####
#####
AHWE2607@ahq e-Sports Club@HongKong Esports@26/07/2015@39:09@ahq
e-Sports Club@20@14@westdoor@#####
#####
PIMY0808@PaiN Gaming@INTZ@08/08/2015@42:55@PaiN Gaming@10@07@Myl
on@#####
#####
JNCH0209@Jin Air Green Wings@NaJin e-mFire@02/09/2015@39:51@Jin
Air Green Wings@08@03@Chei@#####
#####
```

Note que não há quebras de linhas no arquivo (elas foram inseridas aqui apenas para exemplificar a sequência de registros).

Instruções para as operações com os registros:

- **Inserção:** cada partida deverá ser inserido no final do arquivo de dados e atualizado no índice primário.
- **Atualização:** o único campo alterável é o de duração da partida. O registro deverá ser localizado acessando o índice primário e a duração deverá ser atualizada no registro na mesma posição em que está (não deve ser feita remoção seguida de inserção).
- **Remoção:** o registro deverá ser localizado acessando o índice primário. A remoção deverá colocar os caracteres *| nas primeiras posições do registro removido. O espaço do registro removido não deverá ser reutilizado para novas inserções. Observe que o registro deverá continuar ocupando exatamente 192 bytes.

Índices

Um índice primário (Tabela Hash) deve ser criado na inicialização do programa e manipulado em RAM até o encerramento da aplicação. Duas versões de tabelas hash devem ser implementadas, que se diferem na forma de solucionar colisões:

- A versão A aplica a técnica de endereçamento aberto com **reespalhamento linear**;
- A versão B aplica a técnica de **encadeamento**.

Ambas as versões devem armazenar as chaves primárias e os RRNs dos registros. Além disso, a versão A possui um indicador do estado em cada posição (LIVRE, OCUPADO ou REMOVIDO) e a versão B possui um ponteiro para o encadeamento em cada posição.

Deverá ser desenvolvida uma rotina para a criação do índice. A Tabela Hash será sempre criada e manipulada em memória principal na inicialização e liberada ao término do programa.

Para que isso funcione corretamente, o programa, ao iniciar realiza os seguintes passos:

1. Pergunta ao usuário se ele deseja informar um arquivo de dados:
 - Se sim: recebe o arquivo inteiro e armazena no vetor **ARQUIVO**.
 - Se não: considere que o arquivo está vazio.
2. Inicializa as estruturas de dados do índice:
 - Solicita o tamanho e cria a Tabela Hash na RAM;
 - Popula a Tabela Hash a partir do arquivo de dados, se houver.

Interação com o usuário

O programa deve permitir interação com o usuário pelo console/terminal (modo texto) via menu.

A primeira pergunta do sistema deverá ser pela existência ou não do arquivo de dados. Se existir, deve ler o arquivo e armazenar no vetor **ARQUIVO**. Em seguida, o sistema pergunta pelo tamanho da Tabela Hash, que deverá ser sempre um número primo. Você deverá calcular o primeiro primo (T) maior ou igual ao valor informado pelo usuário.

As seguintes operações devem ser fornecidas (nessa ordem):

1. **Cadastro.** O usuário deve ser capaz de inserir uma nova partida. Seu programa deve ler os seguintes campos (nessa ordem): **nome da equipe azul, nome da equipe vermelha, data da partida, duração da partida, nome da equipe vencedora, placar da equipe azul, placar da equipe vermelha e apelido do MVP**. Note que a chave **não** é inserida pelo usuário, você precisa gerar a chave para gravá-la no registro. Você precisa garantir que a data da partida informada esteja no formato “DD/MM/AAAA”, sendo que DD pertence ao intervalo [1, 31], MM pertence ao intervalo [1, 12] e AAAA pertence ao intervalo [2011, 2015]. Você também precisa garantir que a duração da partida é composta por 5 bytes, os placares são compostos por 2 bytes cada, e que o nome da equipe vencedora é igual ao nome da equipe azul ou da equipe vermelha. Caso algum dos campos esteja irregular, exibir a mensagem “**Campo invalido! Informe novamente.**” e solicitar a digitação novamente.
 - **Versão A:** caso a Tabela Hash esteja cheia, exibir a mensagem “**ERRO: Tabela Hash esta cheia!**”. Caso a inserção seja realizada com sucesso, confirmar a inserção e exibir o número de colisões;
 - **Versão B:** as chaves de uma mesma posição devem ser encadeadas de forma ordenada por chave primária. Caso a inserção seja realizada com sucesso, confirmar a inserção.
 - Em ambas as versões, a função de Hash será dada por:

$$h(k) = (1 * k_1 + 2 * k_2 + 3 * k_3 + 4 * k_4 + 5 * k_5 + 6 * k_6 + 7 * k_7 + 8 * k_8) \mod T$$

onde:

k = chave primária com 8 caracteres

k_i = i -ésimo caractere da chave primária

T = tamanho da tabela Hash

2. **Alteração.** O usuário deve ser capaz de alterar a duração de uma partida informando a sua chave primária. Caso ela não exista, seu programa deverá exibir a mensagem “**Registro nao encontrado!**” e retornar ao menu. Caso o registro seja encontrado, certifique-se de que o novo valor informado está dentro dos padrões (5 dígitos MM:SS) e, nesse caso, altere o valor do campo no arquivo de dados. Caso contrário, exiba a mensagem “**Campo invalido! Informe novamente.**” e solicite a digitação novamente.
3. **Busca.** O usuário deve ser capaz de buscar por uma partida informando a sua chave primária. Caso a partida não exista, seu programa deve exibir a mensagem “**Registro nao encontrado!**” e retornar ao menu principal. Caso a partida exista, todos os dados da partida devem ser impressos na tela de forma formatada, exibindo os campos na mesma ordem de inserção.
4. **Remoção.** O usuário deve ser capaz de remover uma partida. Caso ela não exista, seu programa deverá exibir a mensagem “**Registro nao encontrado!**” e retornar ao menu. Para remover uma partida, seu programa deverá solicitar como entrada ao usuário somente o campo chave primária e a remoção deverá ser feita no arquivo de dados com o marcador `*|`.
 - **Versão A:** a posição na tabela Hash deve ser atualizada com o estado **REMOVIDO**;
 - **Versão B:** a chave deve ser removida do encadeamento.
5. **Listagem.** O sistema deverá imprimir a tabela Hash.
 - **Versão A:** Deve imprimir uma posição da tabela por linha, começando pelo índice zero, o estado da posição e a chave correspondente, caso esteja com o estado **OCUPADO**. Por exemplo, considere a Tabela Hash de tamanho 11 a seguir:

```
[0] Ocupado: QLTN3005
[1] Ocupado: SLPY0809
[2] Livre
[3] Livre
[4] Ocupado: CBMA0309
[5] Ocupado: FIRE2204
[6] Ocupado: TRLI0309
[7] Ocupado: QTQG3101
[8] Ocupado: CKNA1309
[9] Livre
[10] Ocupado: PKBR1911
```

- **Versão B:** Deve imprimir uma posição da tabela por linha, começando pelo índice zero, seguido das chaves, se houverem, separadas por um único espaço em branco. Por exemplo, considere a Tabela Hash de tamanho 11 a seguir:

```
[0]
[1]
[2]
[3]
[4] CBMA0309
[5] FIRE2204 TRLI0309
[6]
[7] QTQG3101
[8] CKNA1309
[9]
[10] PKBR1911 QLTN3005 SLPY0809
```

6. **Finalizar.** Libera toda a memória alocada e encerra o programa.

Implementação

Implemente suas funções utilizando como base os códigos fornecidos. Não modifique os trechos de código ou as estruturas já prontas. Ao imprimir alguma informação para o usuário, utilize as constantes definidas. Ao imprimir um registro, utilize a função `exibir_registro()`.

Tenha atenção redobrada ao implementar a operação de listagem da tabela Hash. Atente-se às quebras de linhas requeridas e não adicione espaços em branco após o último caractere imprimível. A saída deverá ser exata para não dar conflito com o **OnlineJudge**. Em caso de dúvidas, examine o caso de teste.

Você deve criar obrigatoriamente as seguintes funcionalidades:

- Criar o índice primário (tabela Hash): deve alocar a tabela de tamanho de um número primo na inicialização do programa;
- Carregar o índice primário: deve construir o índice primário a partir do arquivo de dados;
- Inserir um registro: modificar o arquivo de dados e o índice na memória principal;
- Buscar por registros: buscar por registros pela chave primária;
- Alterar um registro: modificar o arquivo de dados;
- Remover um registro: marcar um registro para remoção no arquivo de dados e remover do índice primário;
- Listar tabela: listar a tabela Hash;

- Finalizar: deverá ser chamada ao encerrar o programa e liberar toda a memória alocada.

Utilizar a linguagem ANSI C.

CUIDADOS

Leia atentamente os itens a seguir.

1. O projeto deverá ser submetido no **OnlineJudge** em dois arquivos diferentes:
 - Para a versão A, reespalhamento linear, arquivo com o nome {RA}_ED2_T03A.c;
 - Para a versão B, encadeamento, arquivo com o nome {RA}_ED2_T03B.c;
2. Não utilize acentos nos nomes de arquivos;
3. Dificuldades em implementação, consultar o monitor da disciplina nos horários estabelecidos;
4. **Identificadores de variáveis:** escolha nomes apropriados;
5. **Documentação:** inclua cabeçalho, comentários e indentação no programa;
6. **Erros de compilação:** nota **zero** no trabalho;
7. **Tentativa de fraude:** nota **zero na média** para todos os envolvidos.