Universidade Federal de São Carlos
Bacharelado em Ciência da Computação
Estruturas de Dados II
Prof. Tiago A. Almeida <talmeida@ufscar.br>

TRABALHO 02 - ÁRVORES-B

Atenção

- **Prazo de entrega: 22/10/2015 – 23h55 (via Moodle).**
  Após o prazo, o trabalho não será considerado!

Indexação usando árvores-B

O sistema de cadastro de partidas profissionais de *League of Legends* foi um sucesso e logo diversos sites decidiram utilizar o sistema. Contudo, com o crescimento acelerado do arquivo de dados, as consultas começaram a ficar lentas e, em consequência, seu programa vem perdendo credibilidade e recebendo uma enxurrada de reclamações dos usuários.

Após analisar o cenário atual, concluiu-se que o uso de índices simples não é mais viável para realizar buscas no arquivo de dados, e que a melhor saída é usar índices de árvores-B para aumentar a eficiência do sistema.

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. Usar 2 campos de 2 dígitos inteiros, 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 da partida** de uma partida a partir da chave primária;
3. Buscar partidas a partir:
   1) da chave primária,
   2) do nome da equipe vencedora, ou
   3) do apelido do MVP.
4. Listar todas as partidas da base ordenadas por:
   1) impressão pré-ordem da árvore-B,
   2) nome da equipe vencedora (ordem lexicográfica), ou
   3) apelido do MVP (ordem lexicográfica).

Dessa vez, nenhum arquivo ficará salvo em disco. O arquivo de dados será simulado em uma *string* e os índices serão sempre criados na inicialização do programa e manipulados 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 com 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).

O arquivo de dados não deverá conter cabeçalho e deverá se chamar `matches.dat`.

Instruções para as operações com os registros:

- **Inserção:** cada partida deverá ser inserida no final do arquivo de dados e atualizada nos índices.
- **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). Note que o campo de duração da partida sempre terá 5 dígitos.

Índices

Três índices deverão ser criados (um de árvore-B e dois índices em listas) na inicialização do programa e manipulados em RAM até o encerramento da aplicação:

- `iprimary`: índice primário (árvore-B), contendo as chaves primárias e os RRNs dos registros. A ordem será informada pelo usuário e **a promoção deverá ser sempre pelo sucessor imediato** (menor chave da sub-árvore direita).
- `iwinner`: índice secundário (lista estática ordenada), contendo o nome da equipe vencedora e a chave primária do respectivo registro, ordenado pelo nome da equipe vencedora.
- `imvp`: índice secundário (lista estática ordenada), contendo o apelido do MVP e a chave primária do respectivo registro, ordenado pelo apelido do MVP.

Deverá ser desenvolvida uma rotina para a criação de cada índice. Os índices serão sempre criados e manipulados em memória principal na inicialização e liberados ao término do programa. Note que o ideal é que a árvore-B (`iprimary`) seja a primeira a ser criada.

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

1. Perguntar 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. Inicializar as estruturas de dados dos índices:
   - Solicitar a ordem m da árvore-B e criar os índices na RAM.

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 pela ordem m da árvore-B usada para indexar as chaves primárias.

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 inválido! Informe novamente:` " e solicitar a digitação novamente.
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 não`

`encontrado!`" e retornar ao menu. Caso o registro seja encontrado, certifique-se de que o novo valor informado está dentro dos padrões (5 bytes) e, nesse caso, altere o valor do campo no arquivo de dados. Caso contrário, exiba a mensagem "`Campo inválido!`" e solicite a digitação novamente.

3. **Busca.** O usuário deve ser capaz de buscar por uma partida:

   - **1. por código:** solicitar ao usuário a 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. Em ambos os casos, seu programa deverá imprimir o caminho percorrido na busca exibindo as chaves contidas nos nós percorridos. **Na última linha do caminho percorrido, adicione uma quebra de linha adicional.**

     Por exemplo, considere a árvore-B de ordem $m = 3$ composta pelos seguintes registros inseridos na ordem `CBMA0309`, `CKNA1309`, `FIRE2204`, `PKBR1911`, `QLTN3005`, `SLPY0809`, `QTQG3101` e `TRLI0309`. (Figura 1), a busca pela chave `CBMA0309` e `FIRE9999` retornará:

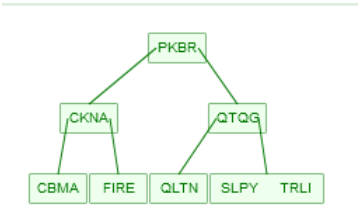

Figura 1: Índice primário estruturado em Árvore-B de ordem 3.

```
Busca por CBMA0309. Nos percorridos:
PKBR1911
CKNA1309
CBMA0309

CBMA0309
CJ Entus
Bangkok Titans
03/09/2013
37:58
CJ Entus
34
13
MadLife

Busca por FIRE9999. Nos percorridos:
PKBR1911
CKNA1309
FIRE2204

Registro nao encontrado!
```

- **2. por nome da equipe vencedora:** solicitar ao usuário o nome da equipe. Caso a equipe informada não tiver ganho nenhuma partida, o programa deve exibir a mensagem "`Registro nao encontrado!`" e retornar ao menu principal. Caso existam uma ou mais partidas cuja equipe vencedora foi a informada pelo usuário, os registros completos dessas partidas deverão ser mostrados na tela de forma formatada, ordenados pela chave primária e separados por uma linha vazia.
- **3. por apelido do MVP:** solicitar ao usuário um apelido. Caso o jogador não tenha sido MVP de nenhuma partida, o programa deve exibir a mensagem "`Registro nao encontrado!`" e retornar ao menu principal. Caso existam uma ou mais partidas cujo jogador informado é o MVP, os registros completos dessas partidas deverão ser mostrados na tela de forma formatada, ordenados pela chave primária e separados por uma linha vazia.

4. **Listagem.** O sistema deverá oferecer as seguintes opções de listagem:

   - **1. árvore-B:** imprime a árvore-B (somente o campo de chave primária) usando varredura **pré-ordem**. Imprimir um nó por linha, começando pelo nível da árvore em que se encontra o nó (a partir da raiz – nível 1) seguido da chave. **Na última linha, adicione uma quebra de linha adicional.** Por exemplo, considere a árvore-B apresentada na Figura 1, a sua listagem resultaria em:

     1 – `PKBR1911`
     2 – `CKNA1309`
     3 – `CBMA0309`
     3 – `FIRE2204`
     2 – `QTQG3101`
     3 – `QLTN3005`
     3 – `SLPY0809`, `TRLI0309`

   - **2. por nome da equipe vencedora:** exibe as partidas na ordem lexicográfica de nome da equipe vencedora.
   - **3. por apelido do MVP:** exibe as partidas na ordem lexicográfica de apelido do MVP da partida.

   As listagens 2 e 3 deverão exibir todos os registros de maneira formatada separados por uma linha vazia. Caso o arquivo de dados esteja vazio, o programa deve exibir a mensagem "`Arquivo vazio!`" e retornar ao menu.

5. **Finalizar.** Encerra a execução do programa. Ao final da execução, feche o arquivo de dados e libere toda a memória alocada pelo programa.

Implementação

Implemente suas funções utilizando como base o código fornecido. 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 as operações de busca e listagem da árvore-B. 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 (árvore-B): deve construir o índice primário a partir do arquivo de dados e da ordem m informada na inicialização do programa;
- Criar os índices secundários (índices simples): deve construir os índices secundários a partir do arquivo de dados;
- Inserir um registro: modificar o arquivo de dados e os índices na memória principal.
- Buscar por registros: buscar por registros pela chave primária ou por uma das chaves secundárias.
- Alterar um registro: modificar o arquivo de dados.
- Listar registros: listar a árvore-B ou todos os registros ordenados por uma das chaves secundárias.
- 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 um único arquivo com o nome `{RA}_ED2_T02.c`, sendo `{RA}` correspondente ao número do seu RA;
2. Não utilize acentos nos nomes de arquivos;
3. Dúvidas conceituais deverão ser colocadas nos horários de atendimento. Dificuldades em implementação, consultar o monitor da disciplina nos horários estabelecidos;
4. **Documentação:** inclua cabeçalho, comentários e indentação no programa;
5. **Erros de compilação:** nota **zero** no trabalho;
6. **Tentativa de fraude:** nota **zero na média** para todos os envolvidos.