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

TRABALHO 01 - INDEXAÇÃO

Atenção

1. **Prazo de entrega: 01/10/2015 – 23h55 (via Moodle).**
   Após o horário limite, o trabalho não será considerado!

2. A atividade deverá ser realizada em duplas.

Organização de arquivos e indexação

League of Legends é um dos jogos online mais jogados no mundo atualmente, com eventos e campeonatos que reúnem milhares de pessoas em grandes estádios. O jogo consiste em duas equipes de cinco jogadores, que se enfrentam com o objetivo principal de invadir as defesas inimigas e destruir a estrutura final, chamada de "*Nexus*".

O *League of Legends World Championship* é o maior evento mundial de *League of Legends* e também um dos maiores campeonatos de *e-sports* do mundo. Em sua 4ª edição, ocorrida em 2014, o evento reuniu 16 dos melhores times do mundo para concorrer a prêmios de até de US$ 1 milhão. A 5ª edição do *League of Legends World Championship* irá ocorrer entre 01/10/2015 e 31/10/2015 e especula-se que haverão prêmios ainda maiores para os competidores.

O dono de um grande portal online sobre *League of Legends* pretende adicionar ao seu site um sistema de bolão, no qual usuários podem concorrer a prêmios tentando adivinhar o resultado do maior número de partidas do campeonato. Você foi contratado para implementar um sistema com algumas funcionalidades para este site. Sua missão é implementar um sistema de cadastro de partidas profissionais de *League of Legends*, para que os usuários do site possam conferir os resultados do bolão. Para cada partida, é preciso armazenar os seguintes dados:

- *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 (*Most Valuable Player*) e do dia e mês da partida (com dois dígitos cada), ex: `FTFE0705` - esse campo é a chave primária, portanto, não poderá existir outro valor idêntico na base de dados;

- *Nome da equipe azul* (nome da equipe que jogou no lado azul do mapa, ex: `Fnatic`);

- *Nome da equipe vermelha* (nome da equipe que jogou 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. Deverá ser igual ao nome da equipe azul ou da equipe vermelha. Ex: `Fnatic`);
- *Placar do jogo* (contagem de abates de cada time com 2 campos de 2 dígitos inteiros cada, ex: `15, 06`);
- *apelido do MVP* (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. Remover uma partida a partir da chave primária;
3. Modificar o campo **duração da partida** de uma partida a partir da chave primária;
4. Buscar partidas a partir:
   1) da chave primária,
   2) do nome da equipe vencedora, ou
   3) do apelido do MVP.
5. Listar todas as partidas da base ordenadas por:
   1) código (ordem lexicográfica),
   2) nome da equipe vencedora (ordem lexicográfica), ou
   3) apelido do MVP (ordem lexicográfica).
6. Liberar espaço.

Para realizar essa tarefa será necessário organizar quatro arquivos distintos: (a) um arquivo de dados que conterá todos os registros, (b) um arquivo de índice primário, (c) um arquivo de índice secundário para o nome da equipe vencedora e (d) um arquivo de índice secundário para o apelido do MVP.

Arquivo de dados

O arquivo de dados deve ser ASCII (arquivo texto) é 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*

da partida (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).

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.
- **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. Além disso, no índice primário, o RRN correspondente ao registro removido deverá ser substituído por -1.
- **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 arquivos com índices deverão ser criados:

- `iprimary.idx`: índice primário, contendo a chave primária e o RRN do respectivo registro, ordenado pela chave primária.
- `iwinner.idx`: índice secundário, contendo o nome da equipe vencedora e a chave primária do respectivo registro, ordenado pelo nome da equipe vencedora.
- `imvp.idx`: índice secundário, contendo o apelido do MVP e a chave primária do respectivo registro, ordenado pelo apelido do MVP.

Deverá ser criada uma rotina para a criação de cada índice. Os índices serão criados carregando para a memória principal os dados necessários, procedendo a ordenação em memória principal e a seguir gravando o arquivo de índice ordenado. Note que o ideal é que `iprimary.idx` seja o primeiro a ser criado.

Ao iniciar o programa, esse deverá carregar os índices para a memória principal e durante a execução do programa, manipular os índices na RAM. Ao fechar o programa, os arquivos de índices deverão ser atualizados no disco.

Para que isso funcione corretamente, o programa, ao iniciar deverá:

1. Verificar se existe o arquivo de dados.
   - Se existir: abrir para escrita e leitura e passar para o item 2.
   - Se não existir: criar o arquivo de dados no disco, abrindo para escrita e leitura.
2. Verificar se existem os arquivos de índices:
   - Se existirem: verificar se estão consistentes com o arquivo de dados (usar uma *flag* para isso).
     i Se estiverem consistentes: carregar os índices para a RAM.
     ii Senão: refazer os índices na RAM e gravá-los no disco.
   - Se não existirem: criar os índices na RAM e gravá-los no disco.

Interação com o usuário

O programa deve permitir interação com o usuário pelo console/terminal (modo texto) via menu. 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. Se a chave primária gerada já existir no arquivo de dados, a seguinte mensagem de erro deverá ser impressa: "`ERRO: Já existe um registro com a chave primária AAAA9999.\n`", onde `AAAA9999` corresponde à chave primária do registro que está sendo inserido.

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. **Remoção.** O usuário deve ser capaz de remover uma partida. Caso ela não exista, seu programa deverá exibir a mensagem "`Registro não 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 por um marcador.

4. **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 não encontrado!`" e retornar ao menu principal. Caso a partida exista, todos os seus dados devem ser impressos na tela de forma formatada, exibindo os campos na mesma ordem de inserção.
   - **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 não 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 não 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.

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

   - **1. por código:** exibe as partidas na ordem lexicográfica de código.

- **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.

A listagem deverá exibir todos os registros (exceto os marcados como excluídos) 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.

6. **Liberar espaço.** O arquivo de dados deverá ser reorganizado com a remoção física de todos os registros marcados como excluídos e os índices deverão ser atualizados.
7. **Finalizar.** Atualiza os índices no disco, fecha os arquivos, libera memória alocada e encerra a execução do programa.

Implementação

Implementar uma biblioteca de manipulação de arquivos para o seu programa, contendo obrigatoriamente as seguintes funcionalidades:

- Uma estrutura de dados para armazenar os índices na memória principal;
- Verificar se o arquivo de dados existe;
- Verificar se o índice primário existe;
- Verificar se os índices secundários existem;
- Criar o índice primário: deve refazer o índice primário a partir do arquivo de dados e substituir, caso haja, o índice existente no disco;
- Criar os índices secundários: deve refazer os índices secundários a partir do arquivo de dados e substituir, caso haja, os índices existentes no disco;
- Carregar os índices do disco para a memória principal;
- Inserir um registro: modificando o arquivo de dados no disco, e os índices na memória principal.
- Buscar por registros: busca pela chave primária ou por uma das chaves secundárias.
- Alterar um registro: modificando o arquivo de dados no disco.
- Remover um registro: modificando o arquivo de dados no disco e o índice primário na memória principal.
- Listar registros: listar todos os registros ordenados pela chave primária ou por uma das chaves secundárias.
- Liberar espaço: removendo fisicamente do arquivo de dados todos os registros marcados como excluídos, e atualizando os índices.

- Atualizar todos os índices: deverá ser chamada ao finalizar o programa e deverá gravar os arquivos de índices no disco a partir das estruturas da memória principal.

Utilizar a linguagem `ANSI C`. Separar a interface da implementação (arquivos `c` e `h`), e fornecer um `Makefile` para compilar o código e retornar ao estado inicial.

A implementação do seu trabalho deverá **obrigatoriamente** ser composto por três arquivos: uma rotina principal chamada `main.c`, um arquivo com as implementações de todas as funções chamado `{RA1}_{RA2}_ED2_T01.c`, cujas definições deverão estar no arquivo `{RA1}_{RA2}_ED2_T01.h`.

CUIDADOS

Leia atentamente os itens a seguir.

1. O projeto deverá ser enviado pelo Moodle observando as seguintes regras:
   - **Apenas um membro do grupo deverá submeter o trabalho.**
   - Submeter um arquivo `zip` chamado `{RA1}_{RA2}_ED2_T01.zip`. Tal arquivo deverá conter uma pasta chamada `{RA1}_{RA2}_ED2_T01` com todos os arquivos que compõem o programa e o `Makefile`.
   - um arquivo texto chamado `{RA1}_{RA2}_ED2_T01_MEMBROS.txt` com o RA e o nome dos membros do grupo em ordem alfabética (cada um em uma linha).
2. Não utilize acentos nos nomes de arquivos;
3. **Identificadores de variáveis:** escolha nomes apropriados;
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 envolvidos. Enfatizo que a detecção de cópia de parte ou de todo código-fonte, de qualquer origem, implicará na reprovação direta na disciplina. Partes do código cujas **idéias** foram desenvolvidas em colaboração com outro(s) aluno(s) devem ser devidamente documentadas em comentários no referido trecho. O que **NÃO** autoriza a cópia de trechos de código nem a codificação em conjunto. Portanto, compartilhem idéias, soluções, modos de resolver o problema, mas não o código.